

웹해킹과 보안

기말고사 대체 과제

학과: 컴퓨터공학과

학번: 18011549

이름: 박태정

**목차**

1. XXE
   1. 참고 할 사항
   2. 문제 풀이
   3. 최근 관련 침해 사례
2. Broken Access Control
   1. 참고 할 사항
   2. 문제 풀이
   3. 최근 관련 침해 사례
3. XSS
   1. 참고 할 사항
   2. 문제 풀이
   3. 최근 관련 침해 사례
4. ‘웹 해킹과 보안’ 수강을 마치며.
5. 참고자료
6. XXE(XML External Entity)

**참고할 사항**

XML Entity 는 반복적으로 나오는 문자열이나 특별처리가 필요한 특수문자를 XML 문서에서 사용하기 위해 미리 정의 해 놓고 사용하는 객체로 & 문자로시작하고 ; 문자로 끝난다.

XXE는 2017년 발표 된 OWASP Top 10 중 4 번째를 차지하고 있는 신흥강자이다. XML 은 업로드가 가능하며, XML 문서에 취약한 코드 등 악의적인 내용을 포함할 수 있다면 공격자는 취약한 XML 프로세스를 공격할 수 있다. 대표적으로 URL 에 대해 외부 객체를 허용하는 경우 발생 가능한 취약점으로 데이터 가져오기, 원격 실행, DOS 공격등 다른 공격들의 실행을 위해 사용 된다.

External Entity 를 이용하여 HTML 패킷에 포함되어있는 XML 객체의 조작이 가능한데, 서버에서 데이터를 가져오는 것이 가능하다. 

XML 파일 외부에 있는 엔티티라도 다음과 같은 방식으로 불러올 수 있게 된느 것이다.

사용자로부터 어떤 인풋을 가져온다면, 프록시를 이용하여 패킷을 가로챈 후

<!DOCTYPE taejeong [

<!ENTITY taejeong SYSTEM “file:///~~]>

& taejeong

이런식으로 XML 을 조작하여 해당 서버에서의 파일 리스트를 쭉 뽑아낼 수도 있다. 하지만 몇가지 조건이 존재한다.

1. 공격자는 XML Request 의 DTD [[1]](#footnote-1)선언을 할 수 있어야 한다.
2. 외부 리소스는 DTD 문법에 맞지 않으면 실행되지 않는다.
3. Binary 는 불러 올 수 없다.

이번 챕터는 오류가 잦아, Mac OS 와 window 7 (VM ) 을 돌아가며 사용했습니다.

Blind XXE

Blind XXE는 공격자가 공격이 성공했는지 실패했는지를 모르는 상태이다. 실제 XXE 에 노출이 되었음에도 외부로 어떤 반응을 하지 않는 경우 생긴다. 결국 공격자입장에서는 일반적인 XXE 보다 XXE 취약점을 응용하기가 어려워 지는 것이다.

일단 test.dtd 파일은 만들고 WebWolf 에 업로드해보겠다. 이때 파일의 내용은 WebWolf Server 로 ping 을 보내는 코드를 작성한다.

스크린샷이(가) 표시된 사진

자동 생성된 설명 스크린샷이(가) 표시된 사진

자동 생성된 설명

WebWolf 에 업로드까지 했다. 그리고 이제 WebGoat 에서 이를 호출하는 코드를 작성한다. 코드는 위에서 작성했던 것 처럼 코멘트를 작성하는란에 burpsuite 를 이용하여 xml 코드를 수정한 후 WebGoat 서버로 보내도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Webgoat 에서 webwolf 에 접근한 것을 로그를 통해 볼 수 있다. 이 말은 공격에 성공한 것인지, 알 수 있다는 것이다. 즉 blind xxe 의 약점을 극복했다고 할 수 있다.

**문제 풀이**

**4-8절**

스크린샷이(가) 표시된 사진

자동 생성된 설명

첫번째 문제는 사진에 코멘트를 남길 때 , file system 의 root directory 를 가져오는 XML 인젝션을 실행하는 것이다. Burpsuite 를 이용해서 XML 을 조작해보도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

기존의 POST 방식의 패킷이다. 단순이 HI 라는 text 만 서버로 전달 되고 있는 상황이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

상기의 패킷 XML 내용을 다음과 같이 바꾸어준다. root 까지 거슬러 올라간 뒤 XXE ENTITY 로 출력을 한다는 의미이다.

그리고 Forward 를 입력하여 server 로 변경 된 패킷을 전송하였다.

****

댓글에 정상적으로 file 들이 출력되는 것을 볼 수 있다.

4-8절

스크린샷이(가) 표시된 사진

자동 생성된 설명

다음은 Modern REST framework 이다. 최신의 REST Frame 워크를 사용한다면 , 개발자가 생각하지 못한 데이터의 형식을 받아들이는 경우도 있다. 그렇게 때문에 **JSON 객체를 받아 처리하는 부분은 XXE 공격에 취약할 수도 있다**. 따라서 Modern REST 로 웹 서비스를 개발한다면 , 클라이언트와 서버가 JSON 객체를 사용할 수도, 혹은 의도하지 않았던 다른 객체로 요청 메시지가 와도 서버에서 그것을 받아들일 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

도저히 모르겠어서 Hint 를 살펴보았다. Content-type 을 살펴보라는데, 확인 해보도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

다음은 댓글을 입력하고 hi 라는 TEXT 가 서버에 전송되는 패킷을 가로챈 것인데, content-type 을 보면, json 인것을 확인할 수 있다.

Burpsuite 에게 json 객체 또한 바꿀 수 있도록 힘을 실어주자.

스크린샷이(가) 표시된 사진

자동 생성된 설명

클라이언트가 요청하는 intercept 에 Json 객체도 포함 시켰다. 이로써 Json 객체도 burpsuite 안에서 수정할 수 있다.

다시 문제에서 댓글 요청을 보내고, Content-type 을 Json -> xml 로 바꾸도록 하겠다.

**스크린샷이(가) 표시된 사진

자동 생성된 설명**

이렇게 수정한 후 Forward 해보겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

요청한 페이지에 루트의 파일이 나온다. 성공한 것 이다.

요약하자면 Json 객체를 받으려고 했던 서비스에서 XML 객체르 보내도 작동을 한다는 것이다. 하지만 그것은 분명 개발자가 의도한 상황은 아니였을 것이다.

XXE 7-8절

스크린샷이(가) 표시된 사진

자동 생성된 설명

XXE 의 마지막 문제이다

이번 섹션에서는 업로드 할 DTD 를 만드는 것이다. **WebGoat server 에서 WebWolf 서버로 dtd 파일을 보내고 댓글을 통해 그것을 호출하는 것이다.** 이 DTD 는 WebWolf 즉 공격자의 서버에서 통제하기 때문에 이 특징을 공격에 활용할 수있게 되는 것이다.

우선 문제에서 제시한 것 처럼 dtd 파일을 생선한다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

Open ENTITY 는 외부 엔티티에면서도 선언 할 떄 % 를 사용하는 파라미터 엔티티이다. 이 % (파라미터 엔티티)는 그 아래 unzip Entity 에서 사용 된다.

Open entity 가 xml 에 의해 해석되고, secret.txt .의 내용이 webwolf 의 렌딩페이지에 기록되어야 하지만 사실은 %open 이 엔티티 참조 없이 그대로 들어간다. 그렇기 때문에 unzip 을 사용하여 run 엔티티로 %open 이 **정상적으로 객체를 참조할** 수 있도록 작성한 것이다.

위에서 말한것과 마찬가지로 , attack.dtd 파일을 만든 것으로 webwolf 에 업로드를 해준다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그리고 해당 코멘트를 보내는 패킷 body 에 다음과 같은 xml 을 넣어준다.

**스크린샷, 그리기이(가) 표시된 사진

자동 생성된 설명**

그리고 패킷을 보내면 WebWolf 에서 문제에서 원하는 값을 뿜어낸다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

나같은 경우에는 콘솔에서 그 값을 발견할 수 이었다. Rst =webgoat 8.0 (~~~ ) 였다. 바로 코멘트 창에 그값을 넣어주니 정답임을 알렸다.

스크린샷이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명

Mac , Window 에서 문제를 해결 했습니다. 스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

이렇게 마지막 문제 까지 풀어 보았다.

**관련 침해 사고**

2017-11-17

어도비 제품군, 신규 취약점 발견...실제 침해사고 악용

<https://www.boannews.com/media/view.asp?idx=58093&page=1&kind=1>

해당 취약점은 공격자가 취약점을 악용하여 악성코드 감염 등의 피해를 발생시킬 수 있다. 이와 관련 한국인터넷진흥원 측은 “해당 취약점은 실제 침해사고에 악용된 사례가 있어 이용자들의 각별한 주의와 적극적인 대처가 필요하다”고 당부했다.   
  
이번에 발견된 취약점은

△Adobe Flash Player

△Adobe Photoshop CC

△Adobe Connect

△Adobe Acrobat and Reader

△Adobe DNG Converter

△Adobe InDesign

△Adobe Digital Editions -> Adobe Digital Editions에서 발생하는 취약점[7]은 △정보 노출로 이어질 수 있는 안전하지 않은 XXE 파싱 취약점(CVE-2017-11273)

△Adobe Shockwave Player

△Adobe Experience Manager에서 발생하는 취약점이다.

1. Broken Access Control

**참고 할 사항**

Broken Access Contorl, 직역은 취약한 접근 통제이다. 접근 인증 절차를 무력화 시키는 것은 어떠한 창의적인 방법이라도 가능하다. 현재 있는 좋은 툴들로 접근통제 절차나, 취약점이 존재하는 지에 대해서 알 수 있다.

이 취약점은 공격자가 관리자의 권한을 그대로 가져와, 사용자의 모든 정보에 접근할 수 있게 되고, 사용자를 가장하여 행동할 수도 있게 되기 때문에, 경제적, 사회적으로 큰 문제를 일으킬 수 있는 공격 기법이다.

취약한 접근 제어 유형의 취약점은 파일다운로드, 파일업로드 URL 파라미터 조작 우회 등이 있다.

**파일다운로드**는 SQL 파라미터를 조작하여 루트 파일에 접근하는 공격기법이다.

**파입업로드**는 게시판 업로드 기능에서 파일 확장자에 대한 필터링이 존재하지 않는 경우 발생 가능한 취약점으로 서버 컴퓨터 전체가 장악 당할 수도 있는 공격기법이다.

**URL 파라미터 조작** URL 의 파라미터를 임의로 조작하면서 비밀글이나 접근할 수 없는 서비스에 접근하게 된다.

현재 그 인증절차와 취약점을 검사하는 툴들의 성격을 알아보도록 하겠다.

1. SAST (Static Application Security Testing)는 서비스의 코드를 검사하는 보안 테스트의 유형 중 하나이다. 일반적으로, 작성된 소스 코드나 코드 설계 방식을 검토하여 발생 가능한 보안 결함을 찾아낸다.
2. DAST (Dynamic Application Security Testing)

응용프로그램을 외부에서 테스트하는 블랙박스 보안 테스트 방식으로 응용프로 그램이 실행중일때 검사를 진행하여 공격자와 마찬가지로 해킹을 시도하면서 결함을 찾아낸다.

**문제 풀이**

Broken Access Control - 2/6 절

스크린샷이(가) 표시된 사진

자동 생성된 설명

문제는 간단하다, 사실 문제라고 하기보다, 그냥 다음 단계를 위해서 어떤 걸 할 지 스토리텔링 쯤 되는 파트인 것 같다.

아이디와 패스워드가 각각 tom , cat 이다.

그대로 입력해주면 통과 된다. 아직까지 취약한 점은 보이지 않는다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

Broken Access Control – 3/6절

스크린샷이(가) 표시된 사진

자동 생성된 설명

다음 문제다. 아주 골머리를 썩은 문제이다. 도대체 문제가 뭘 원하는지를 몰랐기 때문인데, 가상머신인 window7 에서는 View Profile 이 작동하지 않았다. 하지만 Host 인 Mac 에서 ViewProfile 을 누르니, 이게 웬걸, 새로운 화면이 등장했다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

User 에 관한 속성이 있는데, 서버와의 응답에서 가려진 부분들을 입력하라는 것이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

패킷에서 가져온, 현재 웹 페이지에서는 보이지 않는 항목들이 포함 되어 있다.

따라서 페이지에서 보이지 않았던 role 과 userID 를 입력해준다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

통과 했다. 이 부분은 추측하건데, URL 로 userID 를 임의조작한 후 접근할 수 없게 하기 위한 보안 장치로 생각이 된다.

Broken Access Control 4/6 절

스크린샷이(가) 표시된 사진

자동 생성된 설명

나의 프로필을 나의 계정이 아닌 다른 방식으로 view 하는 것이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

위에서 본 것 처럼 Tom 의 userID 는 “2342384” 였음을 잘 기억하고 다른 방식을 찾아보도록 한다. 3 절에서 프로필을 보는 함수에 관해서 패킷에 정보가 있었음을 기억한다. 3절에서 프로필을 확인하는 함수를 다시 한번 살펴보겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

해당 함수를 호출하는 패킷이었다. 경로는

스크린샷이(가) 표시된 사진

자동 생성된 설명

다음과 같이 /WebGoat/IDOR/profile/ 이었다. 그리고 username 혹은 userID 로 profile 에 접근하는것이 분명하다. password 로 접근 하진 않을 것 아닌가.

입력란에 다음과 같이 입력 한다. WebGoat/IDOR/profile/2342384

스크린샷이(가) 표시된 사진

자동 생성된 설명

성공적으로 데이터를 불러오며 성공 메시지가 나왔다. 이것은 추후에 SQL 인젝션 으로도 이용될 가능성이 있어 보인다. 튜플 전체가 속성의 이름까지 나타내면서 출력되기 때문에 매우 취약하다고 할 수 있다.

Broken Access Control 5/6절

스크린샷이(가) 표시된 사진

자동 생성된 설명

Broken Access Control Insecure Direct Object References 파트 마지막 문제이다. 어디서 많이 본 느낌인데, 다른 사람의 프로필을 조회하거나 수정하라는 것이다. 예측컨데, ViewProfile 과 EditProfile 모두 userID 로 그 기능을 조회할 것이기 때문에 tom 의 userID 인 ‘2342384’ 를 이용하도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

아니나 다를까, 방금 이용했던 profile 자리에 어디서 가져온 참조 변수를 하나 가지고 있다. 저곳에 우리 tom 의 userID 보다 4높은 2342388 을 입력해보도록 할 것 이다.



clear.

스크린샷이(가) 표시된 사진

자동 생성된 설명

마찬가지로 값을 그대로 대응 시켜주는 역할을 하는 코드를 작성했다.

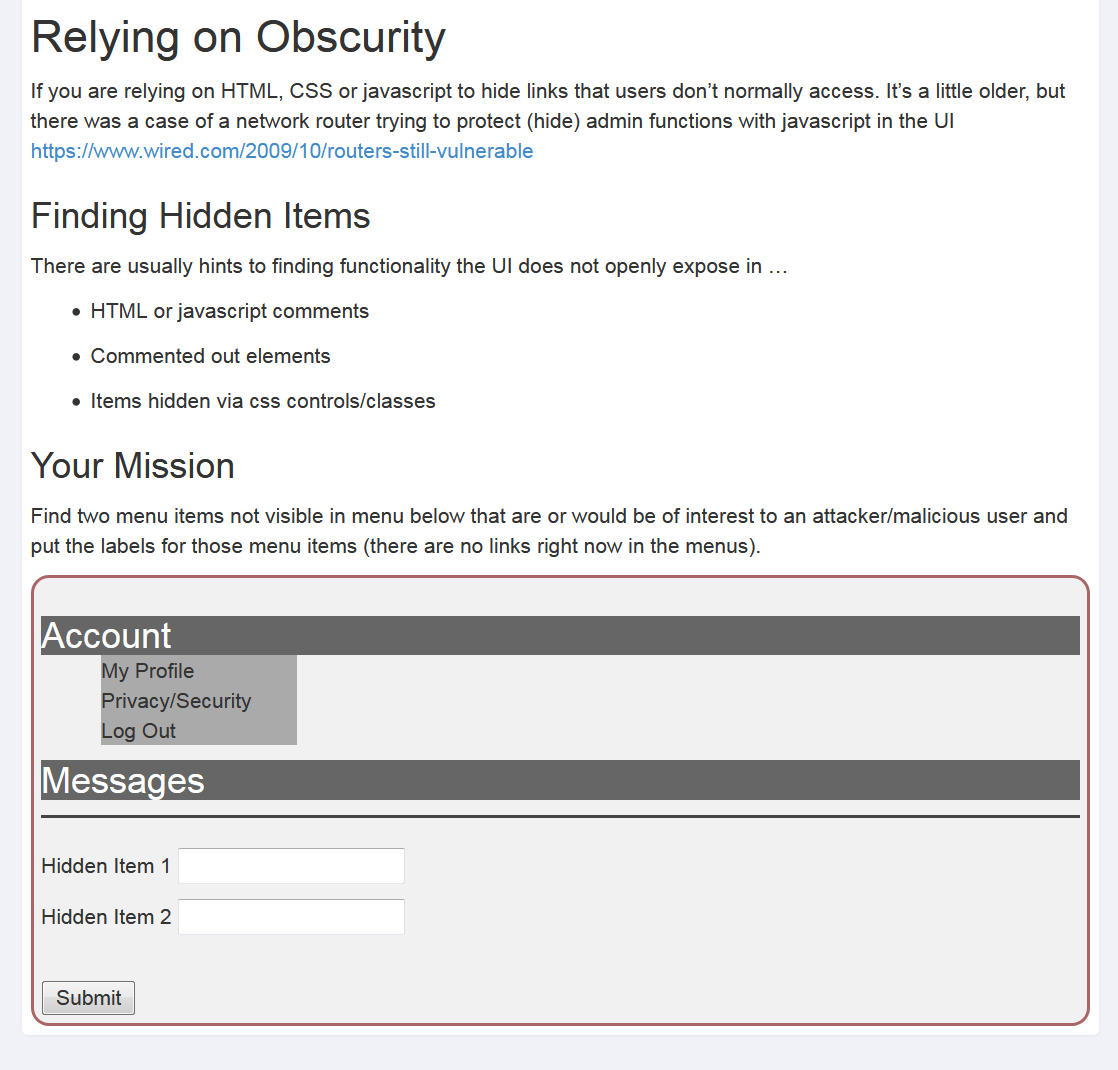
스크린샷이(가) 표시된 사진

자동 생성된 설명

WebGoat가 무언가 제대로 작동하지 않고 있다는 느낌을 강하게 준 챕터였다.

Broken Access Control

Missing Function Level Access Contorl 2 /3절



이번 lesson 에서 말하기를 HTML 에서 링크를 숨기기 위해서 css , javascript 를 사용한적이 있었다고 한다, 그러니까 서버 단이 아니라, 클라이언트단에서 무언가 조치를 취했다고 할 수 있다. 하지만 오래 된 방법이지만 네트워크 라우터에서 javascript 로 admin 을 보호했다고 한다. 이 처럼 웹 서비스안에서 어떤 숨겨진 기능을 찾는것이 이번 lesson 의 목적이라고 할 수 있다.

문맥상 Hidden Item 을 찾는 것이 이번 레슨의 정답이었다. 그래서 firefox 의 Element inspection 기능을 이용해서 hidden 을 검색했더니 href 링크 하이퍼링크 참조로 연결 된 값이 나왔다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

이것을 입력 값에 넣어보도록 하겠다.

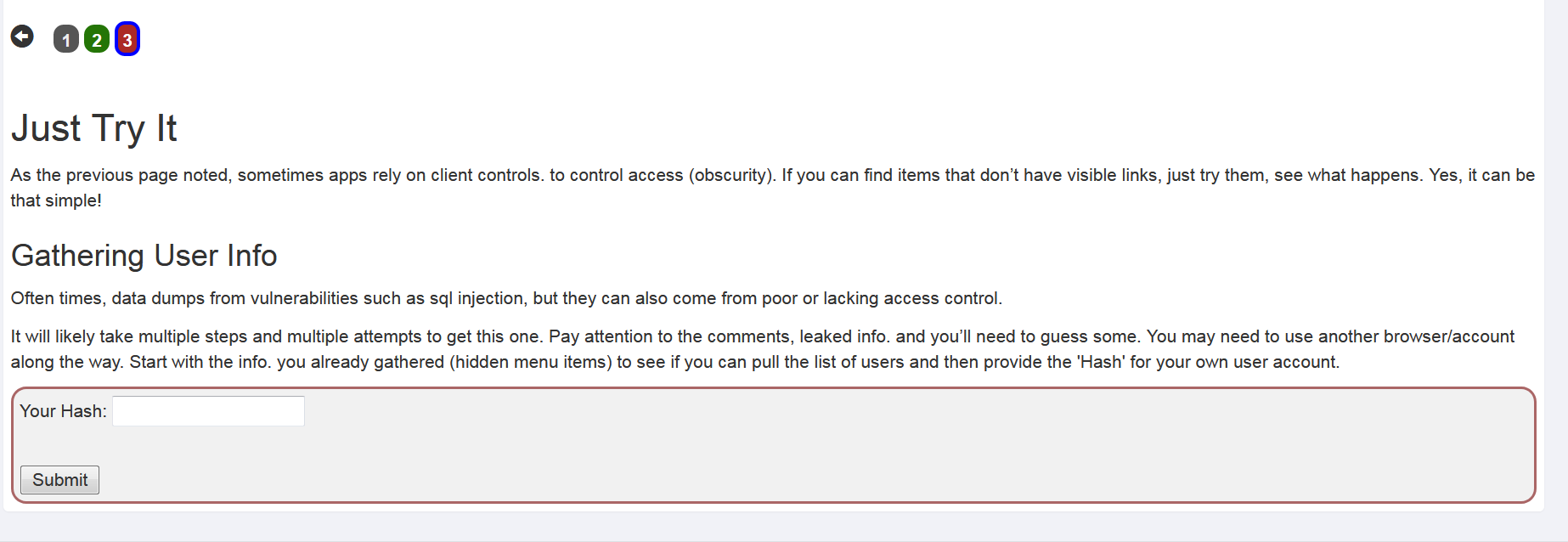
스크린샷이(가) 표시된 사진

자동 생성된 설명

Clear.

Broken Access Control

Missing Function Level Access Contorl 3 /3절



이번 장은 해시를 가져오는 문제이다 . lesson 에서는 sql injection 같은 취약점 때문에 발생하는 데이터 추출은 Access Contorl 의 문제라고 한다. 하지만 이때 데이터를 해시 함수로 처리한다면 그것이 정형화 된 데이터가 아니라면 가져가도 별 의미가 없을 것이다. 사용자의 해시를 찾아보자. 이전 lesson 과 마찬가지로 숨겨진 항목을 이용하면 될 것이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

User 정보로 URL 접근을 시도한다. window 에서 에러가 발생하여 또 mac 으로 이동하였다. 그리고 user 정보에 접근 한 패킷을 수정하여 다시 보낸다. Burpsuite의 repeater 기능을 이용하면 되겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

Repater 를 이용하여 해시를 가져올 수 있다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

Response 에서 얻은 해시로 정답칸에 입력하였다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

이로서 Broken Access Control 도 모두 마무리하였다.

1. XSS(Cross Site Scripting)

**참고 할 사항**

크로스 사이트 스크립팅의 경우 OWASP 에서 제시한 취약점 Top.10 에 항상 있을 정도로 공격 가능성이 높고, 자유롭게 활용할 수 있는 공격 프레임워크의 범위가 넓다. 때문에 대중에게 많이 알려져 있기도 한 공격이다. XSS 는 어쨌든 서버에 접근을 하여야 하기 때문엘 로그를 남긴다. 물론 XSS 와 연계하여 이용하여 제로데이 공격으로 로그 까지 지워버린다면 할 말은 없지만, 로그를 남기기 때문에 탐지가능성이 높은 편에 속한다. 그 종류에는 Reflected XSS , Dom XSS , Stored XSS 가 있는데 이 중 Reflect 와 Dom XSS 의 경우에는 피해정도가 크지 않은 편이다. 하지만 Stored XSS 는 그 피해가 큰 것으로 추정 되고 있다.

XSS , Reflect , Dom , Stored 순서대로 알아 본 뒤 문제를 풀도록 하겠다.

XSS 는 지역 간 , 사용자 간 , 사이트 간 에서 스크립트가 인코딩 혹은 삭제 없이 브라우저로 렌더링 되는 취약점이다. 이에는 검색 문자열을 사용자에게 다시 리턴하는 검색, 혹은 데이터를 반영하는 , 데이터를 반환하는 메시지, 사용자에게 입력을 제공하는 메시지 컨트롤 이나 HTTP Header 등 다양한 섹션에서 공격이 가능하다. HTML 태그 취약점이 대표적이다.

그 종류에는 Reflect ,Dom , Stored XSS (Cross – Site – Scripting )이 있다.

1. Reflect XSS

사용자 요청의 악성코드가 웹 브라우저에서 사용자에게 표시 됨.

서버 응답 후 페이지에 악성코드 기록이 됨.

사회공학적 기법이 필요하다.

브라우저에서 사용자로부터 상속 된 권한으로 실행한다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

1. Dom XSS

사용자 요청의 악성코드가 클라이언트 측 스크립트에서 HTML 사용에 이용 됨

브라우저에서 사용자로부터 상속 된 권한으로 실행한다.

Reflected XSS 와 비슷함.

Dom-based XSS 는 Reflect XSS 의 일종이다. 하지만 Dom-Based 는 Server 로는 악성코드를 전송하지 않는 다는 점에서 기존의 Reflect XSS 와의 차이 점이다.

1. Stored XSS

Stored XSS 의 경우에는 웹 어플리케이션 취약점이 있는 서버에 악성 코드를 반 영구적으로 저장해놓는 것이다. 따라서 이 스크립트는 실제 어떤 게시판이나 댓글등에 작성 되어 있는 경우가 많으며, 사용자는 이러한 스크립트를 보지못하고, 악성 코드가 포함 된 페이지를 긁어올 경우 악성 스크립트를 실행할 수 있게 되는 것이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

|  |  |  |  |
| --- | --- | --- | --- |
|  | Reflect | Dom-Base | Stored |
| 공격 방식 | Victim 에게 직접 | Victim 에게 직접 | 서버와의 통신 |
| 피해 정도 | 중간 | 중간 | 심각 |

**문제풀이**

스크린샷이(가) 표시된 사진

자동 생성된 설명

Cross-Site Scriting Part 의 시작이다. A7 의 Cross-Site Scriting 가 무엇이며, 실제로 개발자가 서비스를 구현 하고 서비스하는 것에 있어서 어떻게 사용 되는지 설명한다.

레슨의 **목적은 실습자가 XSS 가 어떻게 작동하는지 알게 하는 것이다**. 즉, 작동 원리에 대해서 배우게 된다.

Reflected XSS 에 대해서 알게 되며, XSS 기법인 Reflecting XSS 와 DOM-XSS (Stored XSS) 에 대해서 배우게 된다. 바로 문제를 풀어 보도록 하겠다.

XSS 2/12절

스크린샷이(가) 표시된 사진

자동 생성된 설명

편의상 절2개의 레슨 중 2번 째 레슨을 2 / 12 로 부르도록 하겠다. XSS 에 관한 설명이 있다. 문제에 관한 Review 와 짚고 넘어가야 할 점은 다음 파트에서 다루도록 하고, 이 파트에서는 문제를 푸는 것과, 어떻게 푼 것 인지에 대해서 언급하고 넘어가도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

직접 문제를 풀게 되는 공간은 이 Submit Section 이 아니다. Address 입력창에 java script 를 작성하는 것인데, reflect XSS 를 의도하는 것 같다. 쿠키 정보를 가져오는 script 를 실행시키면 되는데, 다른 탭에서도 같은 쿠키를 출력 하느냐를 묻는 문제이다. 당연히 같은 접속 상태로 요청을 한다면 쿠키는 같을 것이다. 뒤에서 쿠키[[2]](#footnote-2)에 대해서 다루도록 하겠다. 그렇다면 요청한 대로 주소창을 이용하여 cookie 를 가져오도록 하겠다.

스크린샷, 테이블, 조류이(가) 표시된 사진

자동 생성된 설명

javascript:alert(document.cookie);

Address Browsing 을 이용하여 쿠키를 가져왔다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

다른 탭에서도 Cookie 를 얻어왔다. XXE 탭에서도 마찬가지로 같은 쿠키를 출력하는 것을 볼 수 있다. 주목할 점은 같은 쿠키 정보라는 것인데 **한 연결에서 같은 쿠키를 내보내는 것이 어떤 의미인지** 뒤에서 살펴보도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

YES 를 입력하고 넘어가도록 하겠다.

XSS 7-12 절

스크린샷이(가) 표시된 사진

자동 생성된 설명

7 – 12 절에서는 Reflected XSS 를 실행시키라고 한다. 여러개의 입력필드에서 XSS 가 작동하는 입력 필드를 찾는다 이번 lesson 에서는 입력 값 검증의 필요성에 대해서 말하려는 듯 하다. 또한 간단하게 XSS 가 작동하는지 알아보기 위해서 간단한(alert, console.log) 등의 스크립트를 입력하곤 한다.

문제를 풀어보겠다. 하나씩 입력해보는 수 밖에 없을 듯 하다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

성공하긴 했는데 , 무언가 다른 스크립트를 원하는 것 같다. 좀 더 Fancy ? 한 작업을 원한다는데 , 이 말이 참 웃긴 것 같다. Fancy 를 원하십니까.

그래서 window.open 과 document.cookie 의 기능을 사용했다. 하지만 그럼에도 불구하고 더 Fancy 한걸 찾길래 그냥 넘어 가기로 했다. 이 Lesson 에서는 클라이언트 단에서 작동하는 Reflect XSS 에 대해서 알아보았다.



10 – 12 절로 넘어가도록 하겠다.

XSS 10-12 절

스크린샷이(가) 표시된 사진

자동 생성된 설명

잠재적인 Dom-based XSS 의 실행이다. 뒤에서 Dom-based XSS 에 대해서 알아보도록 하고 문제를 먼저 풀도록 하겠다.

url 에서 경로를 재구성하여 원하는 class 에 접근하도록 한다. 문제에서 원하는 접근은 “test” 라는 어떤 코드를 찾는 것이다. 이 test code 는 handler 의 경로에 있을 것인데, javascript 라이브러리가 포함되어있는 경로이다. 그렇다면 javascript 가 포함되어있는 경로에서 test 를 호출하면 되는 것이 아닌가 ? 실제로 지금 있는 탭인 이 lesson 에서도 <http://127.0.0.1:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/9>

다음과 같은 경로를 갖게 된다. Lesson 에 직접 들어가게 되는 스크립트들은 lesson section 에 존재하게 될 것이고 실제 test 를 했던 source 는 test section 에 존재하게 될 것이다. CrossSiteScriptong.lesson 은 현재 페이지를 동적으로 사용하게 하는 스크립트이므로 그 이전 섹션에 가서 test 를 시도해보도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

그렇다 test 와 lesson section 은 서로 분리되어 있었고, test 의 어떤 코드들에 접근하는 디렉토리는 lesson 과 같은 level 의 section 인 start.mvc#lesson/ start.mvc#test/ 였다. 뭐 여기서 다른 무언가는 없는 것 같다. 아마도 다음 문제를 위한 단계 였던 것을 보여진다.

바로 다음 문제로 넘어 가도록 하겠다,

XSS 11-12 절

스크린샷이(가) 표시된 사진

자동 생성된 설명

Dom-Based XSS 를 시도한다. 보통 어떤 공격들은 blind 이다. 그러니까 공격이 성공한건지, 실패한건지 모르는 경우들이 있는데, webgoat 의 경우에는 우리의 서버이므로 확인할 수 있다. 뒤 레슨에서 찾은 경로로 함수를 호출하는 것이다. 콘솔로 바로 할 수 있지만 lesson 에서는 url 을 이용하라고 한다. 이후 콘솔에서 번호를 가져오라고 한다. 바로 한번 해보겠다.



Safari 에서

127.0.0.1:8080/WebGoat/start.mvc#test/webgoat.custonjs.phoneHome()

다음과 같은 경로를 입력했다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

함수 호출은 정상적으로 이루어진 것 같은데 , 콘솔에 아무런 반응이 없다. 사실 이것은 아무런 반응이 없는 것이 아니라. 함수 호출을 하지 않은 것이다. 다만, URL 상 test 가 있다면 해당 URL 을 화면에 뿌려주는 작동을 하는 코드가 존재한다.

따라서 이 스크립트, 함수, 클래스를 실행시키기 위해서는 스크립트로 URL 로 보내거나 콘솔을 이용하여 호출을 해야 한다. URL 을 이용해서 이 클래스를 호출해보도록 하겠다.

<http://127.0.0.1:8080/WebGoat/start.mvc#test/javascript:webgoat.customjs.phoneHome()>

<http://127.0.0.1:8080/WebGoat/start.mvc#test/<script>webgoat.customjs.phoneHome()<%2Fscript>> - %2F[[3]](#footnote-3)

두가지 URL 요청을 보냈다. 전자의 경우 마찬가지로 test Handler 만 호출되고 반응이 없어 script 를 직접 작성했다.

스크린샷, 모니터, 화면, 앉아있는이(가) 표시된 사진

자동 생성된 설명

콘솔에서 준 마지막에 보이는 코드를 WebGoat 에 입력해보도록 하겠다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

성공적이다. 다음 단계로 넘어가도록 하겠다.

XSS 12-12 절

스크린샷이(가) 표시된 사진

자동 생성된 설명

마지막 절은 퀴즈이다. 한 문제 씩 짚고 넘어 가도록 하겠다.

첫번째 문제는 신뢰할 수 있는 사이트는 XSS 에 면역이 있는 즉, 영향을 받지 않는 사이트인지에 관한 문제이다.

정답은 4번이다. 문제의 해설에 따라 , 신뢰할 수 있는 것으로 인정이 된다면 해당 스크립트가 악성인지 혹은 실제 순수한 서비스인지 파악 할 수가 없기 때문이다. 따라서 신뢰할 수 있는 사이트라 할지라도 악의적인지 판단이 힘들다.

2번은 언제 XSS공격이 발생하는지에 관한 문제이다

스크린샷, 조류이(가) 표시된 사진

자동 생성된 설명

XSS 공격은 부정적인 성격을 띄는 , 악성코드에 관해 유효성 검사를 하지 않고 웹 사용자에게 전송되는 동적 컨텐츠 (JSP,ASP,PHP) 등에 포함이 된다. 입력으로 클라이언트에서 서버로만 발생이 가능한 것이 아니라 , 서버에서 클라이언트 에게로 공격이 가능 한 이유이다.

3번 문제는 Stored XSS 에 관한 문제이다. 위에서도 살펴 보았듯이 Stored XSS 는 게시글이나 댓글등에 이미 포함 된 스크립트를 User 가 불러올 경우에 발생 가능한 XSS 기법이다.

스크린샷, 조류이(가) 표시된 사진

자동 생성된 설명

정답은 1번이다. 앞서 말했듯이 악성 스크립트는 서버에 지속적으로 저장 되어 있고, 해당 페이지를 서버에 요청하게 된다면 피해자가 악성 스크립트를 읽어들이게 되는 것이다.

4번은 Reflect XSS 관한 문제이다. 스크린샷이(가) 표시된 사진

자동 생성된 설명

Reflect XSS attack 은 웹 서버로의 요청을 보낼 때 Script 를 포함시키는 것을 의미한다. 웹 서버로 전송 된 입력이 요청의 일부일 떄 발생하게 된다.

5번은 XSS공격에 이용 되는 것이 JavaScript뿐이냐고 물어보는 것이다. 명백하게 아니다.스크린샷, 조류이(가) 표시된 사진

자동 생성된 설명

XSS 공격은 Java script 뿐만 아니라 , ASP ,PHP , HTML Tag 등 다양한 웹 서비스 혹은 애플리케이션 단의 어떤 서비스에서 일어나게 된다.

상기의 문제들은 XSS 의 개념을 다시한번 상기시켜주는 용도이다.

이렇게 WebGoat 8.0 XSS 문제를 모두 풀어보았다. 다음은 최근에 발생한 XSS 관련 침해 사고를 알아 보겠다.

**최근 침해 사고**

2017-09-20

북한 추정 해커조직, 포털사이트 다음 취약점 노려 사이버공격

<https://www.boannews.com/media/view.asp?idx=57093&page=1&kind=1>

지난 19일 북한 추정 해커조직에서 국내 대형 포털사이트인 다음 이메일의 img 태그 XSS 취약점을 이용한 공격이 포착됐다. 해당 취약점을 이용해 해커는 메일 로그인 세션 쿠키정보를 탈취하고, 피싱 사이트로 리다이렉션하고 있다.

이번에 발견된 북한 추정 공격은 피싱 화면만 봐서는 일반 피싱 공격과 크게 다르지 않다. 하지만 추적을 피하기 위한 방법과 코딩 등이 이전과는 다른 패턴을 보이고 있는 것으로 분석됐다. 또한 이들의 목적에 따라 다양한 추가 공격이 가능해 지속적인 모니터링이 요구된다.   
  
이에 대해 큐브피아 권석철 대표는 “방어자 입장에서 모든 취약점 패치와 제거는 어려움이 있지만 공격자 입장에서는 한 개의 취약점만 확인이 되면 이를 토대로 자신이 원하는 것을 마음대로 할 수가 있다는 점에서 차이가 있다”며 “결국 해커는 이러한 취약점들을 꾸준히 탐색해 낼 것이며 이를 토대로 국내 특정서버를 명령서버로 계속 활용할 것”이라고 밝혔다.   
  
특히, 이번 사건의 경우 해커가 메일 로그인 세션 쿠키정보를 탈취한다는 점에서 주목되고 있다. 해커가 메일 로그인 세션 쿠키정보를 탈취할 경우 사용자의 메일을 볼 수 있다. 이는 특정인을 타깃으로 정찰과 감시를 통한 정보탈취 목적일 가능성이 높다는 얘기다.

주목할 점은 img 태그 와 해커가 세션 쿠키 정보를 탈취하려고 한다는 것이다.

**Img 태그 취약점**은 HTML 의 이미지 태그에 자바 스크립트를 삽입하는 공격구문이다. 하지만 최근 버전의 업데이트 된 브라우저에서는 동작하지 않는다고 한다. 이미지를 불러오는 주소에 스크립트를 삽입 하는 것이다.

Ex) <IMG SRC = "javascript:alert('you are vulnerable.');">

다음은 **세션 쿠키 정보 탈취에 관한 쟁점이다**. 세션은 한 연결이 생성 될 때 생기게 되는 일종의 cache 이다. 다시 접근하는 사용자의 세션 쿠키를 확인하여 이미 인식 된 **사용자라면 그에 맞는 권한과 서비스를 제공하게 되는 것이다**. 그렇다면 관리자 게정의 쿠키를 빼앗긴다면 어떻게 되는 것일까 ? 해커는 관리자 권한을 가진 계정과 동일한 권한을 가지고 웹사이트를 조절할 수 있게 된다.

이에 관한 공격 기법은 이미 많은 사람들이 알고 있다. 해커는 웹 서버를 세팅하고, cookie 를 가져올 php 파일을 작성한다. 그리고

공격 대상의 입력 파트에 쿠키 정보를 전송하는 스크립트를 작성하고 서버에 저장하게 된다. ( Stored XSS)

스크린샷이(가) 표시된 사진

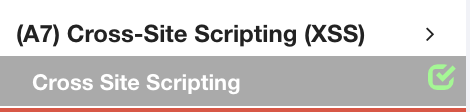
자동 생성된 설명

그리고 admin 이 이 게시글을 보게 된다면 해당 스크립트가 admin 계정에서 실행 될 것이고 해커의 서버인 192.168.52.128 로 admin 의 세션을 전송하게 된다. Admin 입장에서는 스크립트는 보이지 않고 맨 위의 test 만을 보게 될 것이다. 그렇다면 해커의 서버에 어드민의 쿠키가 전송 되었고, 해커는 이를 이용하여 admin 계정을 갖게 되는 것이다.

스크린샷이(가) 표시된 사진

자동 생성된 설명

다음과 같은 프로세스로 작동하게 되는 것이다.



A7 Clear.

수강을 마치며

정말 기대했던 수업인 만큼 웹해킹과 보안 수업은 정말 유익했다. 물론 아쉬운 점은 대면강의가 아니어서 더 적극적으로 수업에 참여하지 못했다는 점인데, 이또한 WebGoat 과제로 적절하게 아쉬운 점을 달랬다고 생각한다. 이번 학기 만큼 과제도 재미있게 한 학기가 없을 것 이다. 과제를 하면서 짜릿함을 느꼈다. 해킹이라는것이 자극적이어서 그렇다고 생각하지 않는다. 실력을 갖추기 위해 필요한 만큼의 인내를 할 생각이고 수업을 듣고 난 후는 어플리케이션 단의 취약점을 직접 분석할 만큼의 실력을 갖출 것이다. 실습도 좋지만 실전에서 사용하지 못하면 의미가 크게 줄어들테니까 말이다. 기대한 것 이상으로 정말 좋은 수업이었다.

한가지 아쉬운 점이 있었다면, 내가 맥북을 사용하고 있었다는 것이다. 가상환경을 만들고, 익숙하지 않은 오류들을 상대하고있으면 하루가 금새 지나있곤 했다. 아무래도 보안 공부에는 윈도우가 제격인 것 같다는 생각을 한다.

자료출처

https://owasp.org/www-project-top-ten/OWASP\_Top\_Ten\_2017/Top\_10-2017\_A7-Cross-Site\_Scripting\_(XSS)

<https://nauri-hacking.tistory.com/55>

<https://www.boannews.com/media/view.asp?idx=57093&page=1&kind=1>

<https://www.youtube.com/watch?v=eTlTAM6oaFg>

https://www.acunetix.com/blog/articles/dast-dynamic-application-security-testing/

1. Document Type Definition 문서의 타입이 이미 선언 된 객체 사용 방법 [↑](#footnote-ref-1)
2. 쿠키 : 인터넷 사용자가 어떠한 웹사이트를 방문할 경우 그 사이트가 사용하고 있는 서버를 통해 인터넷 사용자의 컴퓨터에 설치되는 작은 기록 정보 파일을 일컫는다. [↑](#footnote-ref-2)
3. URL 에서는 특정 특수문자 ‘ / ‘ 슬래시의 경우 세부경로 탐색으로 인식 되므로 다른 방식으로 / 를 사용해야 한다. [↑](#footnote-ref-3)