제14장 응용 보안

1. HTTP

HTTP는 Web상에서 HTML 문서를 주고받을 수 있는 프로토콜이다.
HTTP(HyperText Transfer Protocol)는 응용층 프로토콜이다.
HTTP는 RFC 1945, RFC 2616에 정의되어 있다.
서로 다른 종단시스템(서버, 클라이언트)은 HTTP 메시지를 교환하여 통신할 수 있다.

1 무상태 프로토콜(stateless protocol)

- HTTP 서버는 클라이언트에 대한 어떤 정보도 유지하지 않는다.
- 클라이언트가 객체 재전송을 요청하면 서버는 이전 일을 기억 못하므로 그 객체를 또 보낸다.
- 이러한 HTTP 특성을 **무상태 프로토콜**이라 한다.
- **무상태 프로토콜** : HTTP, 인터넷 프로토콜(IP) 등

2. 웹 페이지(web page) 구성

- 대부분의 웹 페이지(web page)는 다양한 객체(object)들로 구성된다.
- 여기서, 객체(object)는 html 텍스트와 jpeg, gif 등과 같은 이미지를 말한다.
- 웹 페이지가 html 텍스트와 5개의 gif 이미지로 구성 : 6개의 객체를 갖는 웹 페이지
- 각 객체는 단순히 단일 **URL**로 지정할 수 있는 하나의 파일(html, jpeg, gif 등)이다.
- 기본 HTML 파일은 페이지 내부의 다른 객체를 그 객체의 URL로 참조한다.

3. HTTP의 지속연결(persistent connection) / 비지속연결(non-persistent connection)

지속연결	• 요청과 그에 대한 응답이 동일한 연결로 보내지는 연결이다.
비지속연결	• 요청과 그에 대한 응답이 분리된 별도의 연결이다.

- 모든 (요청, 응답) 쌍이 **동일한** TCP 연결을 통해 보내져야 하는가?
- 모든 (요청, 응답) 쌍이 **분리된** TCP 연결을 통해 보내져야 하는가?

[예제] 비지속연결(non-persistent connection)

- 전체 웹페이지 구성 : 기본 HTML 파일과 5개의 JPEG 이미지로 구성
- 기본 HTML 파일의 URL : **http://www.sample.com/root/a.html**
- 기본 HTML 파일은 페이지 내부의 다른 5개의 이미지 객체를 그 객체의 URL로 참조
- 그리고, 6개의 객체가 모두 같은 서버에 있을 경우

↓비지속연결 과정은 다음과 같다.

〈비지속연결〉

① 클라이언트는 포트번호 80을 통해 **http://www.sample.com** 서버로 TCP 연결을 시도한다.
- TCP 연결과 관련하여 클라이언트와 서버에는 각각의 소켓이 있다.

② 클라이언트는 단계 ①에서 설정된 TCP 소켓을 통해 서버로 HTTP 요청메시지를 보낸다.
- 이 요청메시지는 **/root/a.html** 경로 이름을 포함한다.

③ HTTP 서버는 단계 ①에서 연결된 소켓을 통하여 요청메시지를 받는다.
- 저장장치로부터 **/root/a.html** 객체를 추출한다.
- HTTP 응답메시지에 그 객체를 캡슐화한다.
- 그리고, 응답메시지를 소켓을 통해 클라이언트로 보낸다.

④ HTTP 서버는 TCP에게 TCP 연결을 끊으라고 지시한다.**(비지속연결)**
- 실제로는 TCP 클라이언트가 응답메시지를 받을 때까지 연결을 끊지 않는다.

⑤ HTTP 클라이언트가 응답메시지를 받으면, **TCP 연결이 중단**된다.
- 응답메시지는 캡슐화된 객체가 HTML 파일인 것을 나타낸다.
- 클라이언트가 응답메시지로부터 파일을 추출하고 HTML 파일을 조사한다.
- 그리고, 나머지 5개의 JPEG 객체에 대한 참조를 찾는다.

⑥ 그 이후에 참조되는 각 5개의 JPEG 객체에 대하여 **처음 ①에서 ④까지 단계를 반복**한다.

- 서버가 클라이언트에 객체를 보내고 TCP 연결이 끊어지는 비지속연결을 사용하고 있다.
- 서버와 클라이언트 연결은 다른 객체 전송을 위해 연결이 유지되지 않는다.
- 각 TCP 연결은 하나의 요청메시지와 하나의 응답메시지만 전송한다.
- 해서, 주어진 예에서는 사용자가 웹 페이지를 요청할 때 6개의 TCP 연결이 만들어진다.

◈ **지속연결(persistent connection)**
- 지속연결은 서버는 클라이언트에 응답을 보낸 후에 TCP 연결을 그대로 유지한다.
- 같은 클라이언트와 서버 간의 이후 요청과 응답은 같은 연결을 통해 보내진다.
- 전체 웹페이지(예제에서 HTML 파일과 5개의 이미지)를 하나의 TCP 연결을 통해 보낸다.
- HTTP의 디폴트 모드는 **파이프라이닝을 이용한 지속연결**을 사용한다.

4. HTTP 요청/응답 메시지

HTTP는 클라이언트와 서버 사이에 **요청/응답**(request/response) 원리이다.**(80 포트 사용)**

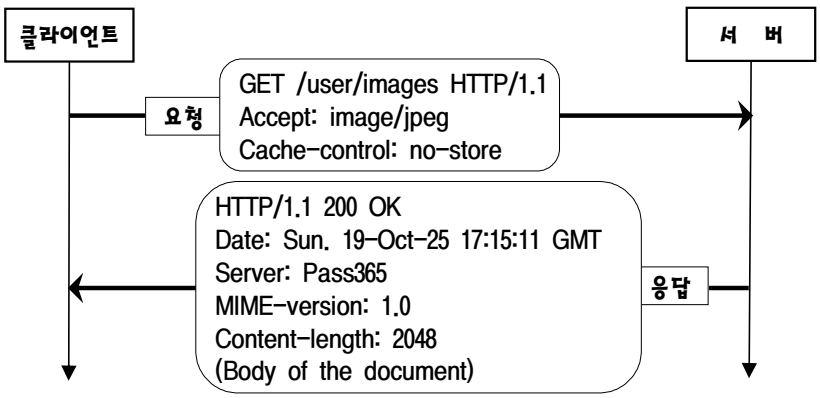
요청 메시지 (클라이언트)

요청 라인(request line)	→ 요청 종류, URL, HTTP 버전 등을 지정
헤더(header)	→ 클라이언트와 서버 사이의 추가적인 정보 교환 (Accept, Referer, **Cookie**, Content-length 등)
공백(a blank line)	→ 한 줄을 띄운다.
본문(body)	→ 전달할 메시지가 있을 때 사용

응답 메시지 (서버)

상태 라인(status line)	→ 응답 메시지 상태(HTTP 버전, 상태 코드, 상태 문구)
헤더(header)	→ 클라이언트와 서버 사이의 추가적인 정보 교환 (Accept, Referer, Cookie, Content-length 등)
공백(a blank line)	→ 한 줄을 띄운다.
본문(body)	→ 전달할 메시지가 있을 때 사용

[예제] HTTP에서 클라이언트/서버 사이의 요청과 응답



GET /user/images HTTP/1.1 → /user/images에 있는 데이터 읽기

Accept: image/jpeg → 클라이언트가 수용할 수 있는 이미지 형식(서버에 알림)

Cache-control: no-store → 수신 내용에 대한 캐시 저장 여부를 결정(서버에 알림)

---

4 http://cafe.daum.net/pass365(홍재연)

◈ HTTP 응답 코드 (일부 내용)

코드	메시지	설명
100	Continue	계속, 나머지 요청 정보를 계속 보내주길 바람
200	OK	오류 없이 전송 성공(서버가 클라이언트에게)
202	Accepted	서버가 클라이언트의 요청을 수락 - 접수
400	Bad Request	요청 실패. 문법 오류로 서버가 이해 못함
401	Unauthorized	권한 없음(인증되지 않았음) - 접속실패
402	Payment Required	예약됨 - 요금 지불 요청
404	Not Found	요청한 문서를 찾을 수 없음(오류 메시지)
500	Internal Server Error	서버 내부 오류
501	Not Implemented	필요한 기능이 서버에 설치되지 않았음 - 구현 불가
502	Bad gateway	게이트웨이 상태 나쁨 - 불량 게이트웨이

• 404의 오류 메시지를 이용하여 악성코드를 삽입할 수 있다.

◈ 요청 메소드 Get과 Post

GET	• Get은 URL 끝에 ?와 함께 (이름, 값) 쌍으로 서버에게 요청을 보낸다. (전송 자료 노출) • 예 1 : www.sample-url.com/resources?name1=value1&name2=value2 • 예 2 : www.example.com?id=pass&pw=365 • Get은 소량의 데이터를 전송할 수 있다. • Get 요청은 서버의 자료나 상태를 변경시키지 않는다. • Get은 서버의 게시판 글을 읽는 등 정보를 조회하는데 주로 사용된다. • Get 요청은 서버에게 여러 번 동일한 요청을 해도 동일한 결과가 돌아온다.
POST	• Post는 HTTP 요청메시지의 본문(body)에 특정 자료를 담아서 서버에게 요청을 보낸다. • Post는 대량의 데이터 전송이 가능하다. • Post 요청은 서버의 자료나 상태를 변경시킬 수 있다. • Post는 자료 생성, 수정, 삭제에 사용될 수 있다. • 하지만, 자료 생성에는 Post를 사용하고 • 자료 수정에는 Put 또는 Patch를 사용하고 • 자료 삭제에는 Delete를 사용하는 것이 바람직하다. • Post 요청은 서버에게 여러 번 동일한 요청을 하면 응답은 서로 다를 수 있다.

• Get은 멱등(idempotent) 성질을 가지도록 설계되었고
• Post는 non-idempotent 성질을 가지도록 설계되었다.

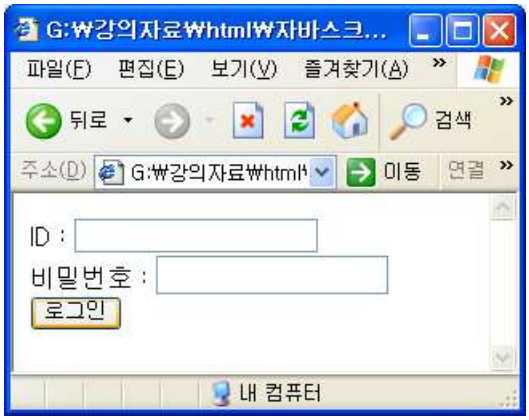
• 멱등은 수학에서 여러 번 연산을 적용해도 그 결과가 달라지지 않는 성질이다.

[예제] 로그인을 처리하는 JavaScript 프로그램

다음은 로그인을 위해 id와 비밀번호를 입력하는 프로그램이다.

```
<html>
  <script language="javascript">
    function nullCheck(){
      var id = document.test.id.value;                    //여기서, test는 form 객체 이름
      var pw = document.test.pwd.value;
      if(!id) alert("아이디(id)를 입력하세요!");
      if(!pw) alert("비밀번호를 입력하세요!");
    }
  </script>
  <body>        //form은 클라이언트(웹브라우저)에서 입력한 자료를 서버로 전달한다.
    <form name="test" action="http://www.naver.com" method="post">        //Post 요청
      ID        : <input type="text" name="id"><br>
      비밀번호 : <input type="password" name="pw"><br>
      <input type="button" name="submit" value="로그인" onClick="nullCheck()">
    </form>
  </body>
</html>
```

- 로그인에서 id와 비밀번호가 입력되었는지를 서버가 아닌 브라우저에서 검사하는 프로그램이다.
- 데이터 전송없이 데이터 입력 여부를 브라우저에서 검사하면 네트웍의 부하를 줄일 수 있다.



- ID를 입력하지 않고 '로그인' 단추를 누르면 위와 같은 안내 메시지가 나타난다.
- 데이터 입력 여부는 웹 브라우저에서 검사된다.

- 메소드 alert() : 다이얼로그 박스로 경고 메시지 등을 출력한다.
- 이벤트 onClick : 마우스로 button, checkbox, radio를 클릭할 때 발생된다.

5. HTTP 관련 공격

① HTTP Get Flooding 공격

- HTTP Get Flooding은 단순하게 **HTTP 요청 공격**이다.
- HTTP Get Flooding은 TCP 연결 이후, HTTP 요청 과정이 수행되는 **DDoS** 공격이다.
- HTTP Get Flooding은 **응용층 취약점을 공격**하는 기법이다.

방어	• 정상적인 HTTP Get 요청인지? 검사해야 한다. • HTTP Get 요청의 임계값을 모니터링 • 비정상적으로 많은 트래픽을 발생하는 출발지 IP에 대한 선별적인 차단

② HTTP CC(cache-control) 공격

- HTTP CC는 **HTTP 요청 헤더의 cache control 필드 값을 조작하는 공격**이다.
- **캐시제어(cache control)** 필드는 HTTP Request 헤더에 있다.

↓캐시제어 필드 값을 **no-store**로 지정하면,**(캐시에 저장하지 않음)**

- 클라이언트 요청에 포함된 어떤 응답 내용도 캐시에 저장되지 않는다.
- 해서, 같은 요청에 대해 서버는 일일이 재전송하므로 서비스 제공에 **부하**가 **증가**하게 된다.
- **결론**은, 캐시제어(cache control) 필드 값을 조작하여 **DoS** 공격이 가능하다.
- no-store는 민감한 정보를 백업하여 보유하거나 배포하는 것을 방지하기 위한 것이다.
- HTTP CC 공격은 서버에 더 많은 부하를 유발시키는 지능화된 **DDoS** 공격이다.
- Get은 클라이언트가 서버로부터 문서 자체를 읽어 오기 원할 때 사용한다.
- Get은 URL에 해당하는 자료의 전송을 요청한다.

③ Slowloris(슬로로리스) 공격

- Slowloris는 **HTTP 서버의 연결을 소진**시키는 **DDoS** 공격이다.
- Slowloris 공격은 **적은 양**의 트래픽으로 공격 가능하다.
- Slowloris 공격은 적은 양의 트래픽을 사용하므로 DoS **공격 탐지**가 어렵다.

공격 원리	• Slowloris 공격은 TCP 세션 연결 후에 GET 요청에서 **불완전한** HTTP 헤드를 보낸다. • **정상적인** HTTP GET 요청 : 마지막은 **2개**의 개행문자(\ r \ n \ r \ n)로 **구성**됨 • **불완전한** HTTP GET 요청 : **2개**의 개행문자 중 **하나**(\ r \ n)를 **제거**한 패킷 • apache 서버는 불완전한 GET 요청을 받으면, 바로 응답하지 않고 300초 기다린다. • 공격자는 시간이 만료되기 전에 다시 불완전한 GET 요청을 보낸다.

- Slowloris 공격에 **윈도** 서버는 취약하지 않았고, **apache** 서버는 취약하였다.
- 윈도 서버는 최대로 연결할 수 있는 개수를 130개로 제한하였으므로

기출문제 분석

1. 다음 중 캐싱(caching) 장비가 응답하지 않도록 설정된 다수의 HTTP GET 패킷을 특정 시스템에 전송하여 서비스를 마비시키는 공격으로 옳은 것은? [2017년 국회 9급]

① Slowloris 공격
② HTTP GET Flooding 공격
③ ARP Spoofing 공격
④ DNS Spoofing 공격
⑤ HTTP CC(Cache-control) 공격

♤ **HTTP CC(cache-control) 공격**

- HTTP CC는 **HTTP 요청 헤더의 cache control 필드 값을 조작하는 공격**이다.
- 캐시제어(cache control) 필드는 HTTP Request 헤더에 있다.
- 캐시제어 필드 값을 no-store로 지정하면,(캐시 사용 안함)
  클라이언트 요청에 포함된 어떤 응답 내용도 캐시에 저장되지 않는다.
  해서, 같은 요청에 대해 서버는 일일이 재전송하므로 서비스 제공에 부하가 증가하게 된다.
- 결론은, 캐시제어(cache control) 필드 값을 조작하여 DoS 공격이 가능하다.

정답 : ⑤

2. 브라우저가 웹 서버로부터 정보를 읽어 오기 위해 사용하는 응용층 프로토콜은? [2016년 지방컴일 9급]

① SMTP ② HTTP
③ IMAP ④ RTP

♤ **통신 프로토콜**

- SMTP : 인터넷에서 메일을 보내기 위한 응용층 프로토콜
- HTTP : WWW 상에서 정보를 주고받을 수 있는 응용층 프로토콜
- IMAP : 원격 서버로부터 TCP/IP 연결을 통해 메일을 가져오는 응용층 프로토콜
- RTP(Real-time Transport Protocol) - 실시간 전송 프로토콜
- RTP는 IP 네트워크를 통해 오디오와 비디오를 전달하기 위한 표준 패킷 포맷을 정의한다.
- RTP는 스트리밍 미디어를 포함하는 통신 시스템에 널리 사용되고 있다.

정답 : ②

3. 웹 서버와 클라이언트 간의 쿠키 처리 과정으로 옳지 않은 것은? [2023년 국가 9급]

① HTTP 요청 메시지의 헤더 라인을 통한 쿠키 전달
② HTTP 응답 메시지의 상태 라인을 통한 쿠키 전달
③ 클라이언트 브라우저의 쿠키 디렉터리에 쿠키 저장
④ 웹 서버가 클라이언트에 관해 수집한 정보로부터 쿠키를 생성

♤ **웹 서버와 클라이언트 간의 쿠키 처리 과정**

• **쿠키**는 HTTP 요청 메시지의 **헤더** 라인을 통해 전달된다.

요청 메시지 (클라이언트)

요청 라인(request line)	→ 요청 종류, URL, HTTP 버전 등을 지정
헤더(header)	→ 클라이언트와 서버 사이의 추가적인 정보 교환 (Accept, Referer, **Cookie**, Content-length 등)
공백(blank line)	→ 한 줄을 띄운다.
본문(body)	→ 전달할 메시지가 있을 때 사용

응답 메시지 (서버)

상태 라인(status line)	→ 응답 메시지 상태(HTTP 버전, 상태 코드, 상태 문구)
헤더(header)	→ 클라이언트와 서버 사이의 추가적인 정보 교환 (Accept, Referer, Cookie, Content-length 등)
공백(blank line)	→ 한 줄을 띄운다.
본문(body)	→ 전달할 메시지가 있을 때 사용

// 쿠키 생성 및 저장

웹 브라우저	요청 → ← 응답(쿠키 생성)	웹 서버	• 쿠키가 웹 브라우저로 전송된다. • 쿠키는 HTTP 요청 메시지 **헤더 라인**을 통해 전달된다..
웹 브라우저 [쿠키]		웹 서버	• 쿠키가 쿠키 저장소에 저장된다.
웹 브라우저 [쿠키]	요청 → [쿠키]	웹 서버	• 같은 웹 서버 접속시 쿠키도 같이 실려감

정답 : ②

4. HTTP 응답 메시지 상태코드의 의미가 옳지 않은 것은? [2020년 국가 7급]

① 201 - Created
② 301 - Moved Permanently
③ 401 - Unauthorized
④ 501 - Bad Request

♤ HTTP 응답코드

- HTTP는 클라이언트와 서버 사이에 **요청/응답**(request/response) 원리이다.**(80 포트 사용)**

◈ HTTP 응답코드 (일부 내용)

코드	메시지	설명
100	Continue	계속, 나머지 요청 정보를 계속 보내주길 바람
101	Switching Protocol	프로토콜 전환, 요청자가 서버에 프로토콜 전환을 요청했다.
200	OK	성공, 오류 없이 전송 성공(서버가 클라이언트에게)
201	Created	작성됨, 성공적으로 요청되었으며 서버가 새 리소스를 작성했다.
202	Accepted	허용됨, 서버가 클라이언트의 요청을 수락 - 접수
300	Multiple Choices	복수 선택, 서버가 요청에 따라 여러 조치를 선택할 수 있다.
301	Moved Permanently	영구 이동, 요청한 페이지를 새 위치로 영구적으로 이동했다.
400	Bad Request	잘못된 요청. 문법 오류로 서버가 이해 못함
401	Unauthorized	권한 없음, 인증되지 않았음 - 접속실패
402	Payment Required	결재 필요, 요금 지불 요청
404	Not Found	찾을 수 없음, 요청한 문서를 찾을 수 없음(오류 메시지)
500	Internal Server Error	서버 내부 오류, 서버에 오류가 발생하여 요청을 수행할 수 없다.
501	Not Implemented	구현 불가, 서버에 요청을 수행할 수 있는 기능이 없다.
502	Bad gateway	불량 게이트웨이, 게이트웨이 상태 나쁨

// HTTP 응답코드는 5개의 클래스(분류)로 구분된다. 첫 번째 숫자는 응답 클래스를 정의한다.
1xx (정보) : 요청을 받았으며, 프로세스를 계속한다.
2xx (성공) : 요청을 성공적으로 받았으며, 인식했고 수용하였다
3xx (리다이렉션) : 요청 완료를 위해 추가 작업 조치가 필요하다
4xx (클라이언트 오류) : 요청의 문법이 잘못되었거나 요청을 처리할 수 없다
5xx (서버 오류) : 서버가 명백히 유효한 요청에 대해 충족을 실패했다

정답 : ④

5. HTTP 버전 1.1에 대한 설명으로 옳지 않은 것은? [2019년 국가 7급]

① TCP를 전송 프로토콜로 사용한다.
② 요청 메시지의 첫 줄인 요청 라인에는 메소드, URL, HTTP 버전 필드가 포함된다.
③ 요청과 그에 대한 응답이 같은 연결로 보내지는 지속연결(persistent connection)을 기본으로 하며, 분리된 별도의 연결을 이용하는 비지속연결(non-persistent connection)도 지원한다.
④ HTTP 서버가 클라이언트에 대한 정보를 유지하는 상태(stateful) 프로토콜이다.

♤ **HTTP 버전 1.1**

- HTTP 서버가 클라이언트에 대한 정보를 유지하는 **상태**(stateful) 프로토콜이다.**(×)**
  → HTTP는 서버가 클라이언트에 대한 정보를 유지하지 못하는 **무상태** 프로토콜이다.
- **무상태 프로토콜**(stateless protocol) : HTTP, 인터넷 프로토콜(IP) 등

정답 : ④

6. 다음에서 설명하는 보안 공격은? [2022년 지방 9급]

○ 정상적인 HTTP GET 패킷의 헤더 부분의 마지막에 입력되는 2개의 개행 문자(\r\n\r\n) 중 하나(\r\n)를 제거한 패킷을 웹 서버에 전송할 경우, 웹 서버는 아직 HTTP 헤더 정보가 전달되지 않은 것으로 판단하여 계속 연결을 유지하게 된다.
○ 제한된 연결 수를 모두 소진하게 되어 결국 다른 클라이언트가 해당 웹 서버에 접속할 수 없게 된다.

① HTTP Cache Control
② Smurf
③ Slowloris
④ Replay

♤ **HTTP – Slowloris(슬로로리스) 공격**

- Slowloris는 **HTTP 서버의 연결을 소진**시키는 DDoS 공격이다.
- 윈도 서버는 최대로 연결할 수 있는 개수를 130개로 제한하였으므로
- Slowloris 공격은 TCP 세션 연결 후에 GET 요청에서 **불완전한** HTTP 헤드를 보낸다.
  → **불완전한** HTTP 헤드 : **2개**의 개행 문자(\r\n\r\n) 중 **하나**(\r\n)를 **제거**한 패킷
  → apache 서버는 불완전한 GET 요청을 받으면, 바로 응답하지 않고 300초 기다린다.
  → 공격자는 시간이 만료되기 전에 다시 불완전한 GET 요청을 보낸다.
- Slowloris 공격은 적은 양의 트래픽으로 공격 가능하다.
- Slowloris 공격은 적은 양의 트래픽을 사용하므로 DoS 공격 탐지가 어렵다.

정답 : ③