12장 - 기본 인증 (Basic Authentication)

인증이란 누구인지 증명하는 것.

웹에서 공용이 아닌 정보를 다루기 위해 허가된 사람만 데이터에 접근하고 처리할 수 있는 기능(**인증**)이 필요합니다.

-> 서버가 사용자가 누구인지 식별

-> 사용자가 어떤 리소스, 작업에 접근 가능한지 결정할 수 있음

HTTP는 자체적인 인증 관련 기능을 제공함

> 사실 대부분의 웹사이트는 보안을 더 강화하기 위해 인증 모듈을 이용해 직접 구현함.

이 장에서는

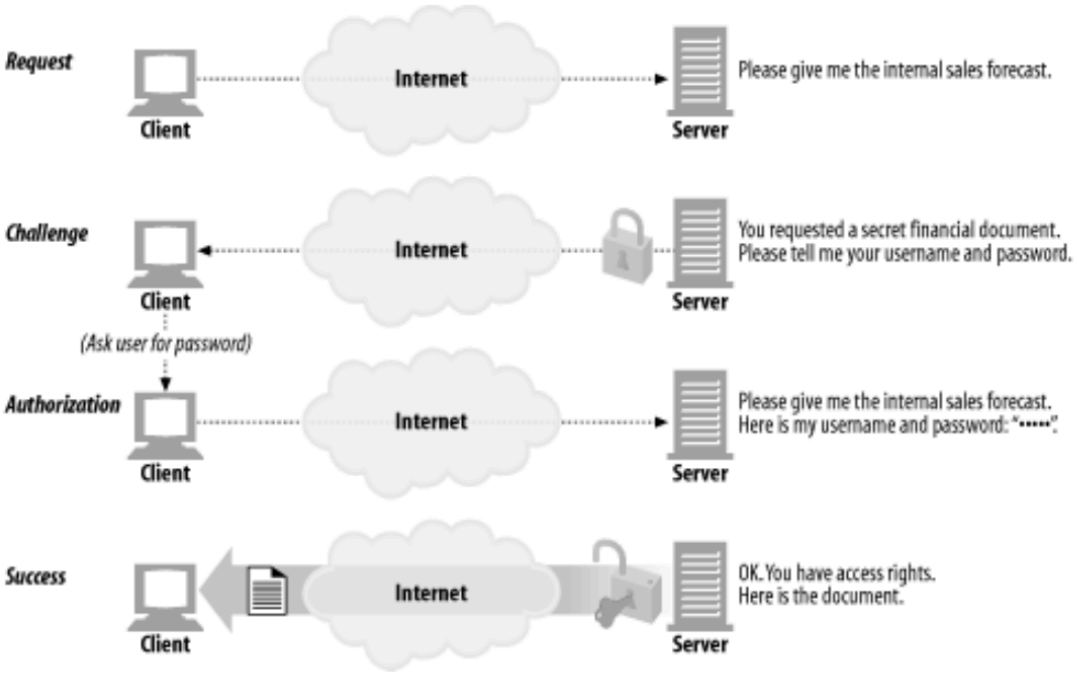
1. HTTP 인증
2. 기본 인증

에 대해 설명합니다.

HTTP의 인증요구/응답 프레임워크

HTTP는 자체 인증요구/응답 프레임워크를 제공함

아래의 그림은 인증 모델



1. 서버에서 클라이언트로부터 **http요청을 받음**
2. 요청 처리 전에 현재 사용자가 누구인지 클라이언트에게 요구(**개인 정보 요구/인증 요구**)
3. 클라이언트는 **인증 정보**(이름, 비밀번호)를 첨부하여 **다시 요청**

4. 정보가 맞지 않으면 **다시 요구하거나 에러표시** / 맞으면 **요청 처리**

인증 프로토콜과 헤더

HTTP에는

- 기본 인증
- 다이제스트 인증

위 두가지 공식 인증 프로토콜이 있는데, 여기서는 기본 인증에 대해서만 다룹니다. (다이제스트 인증은 13장에서)

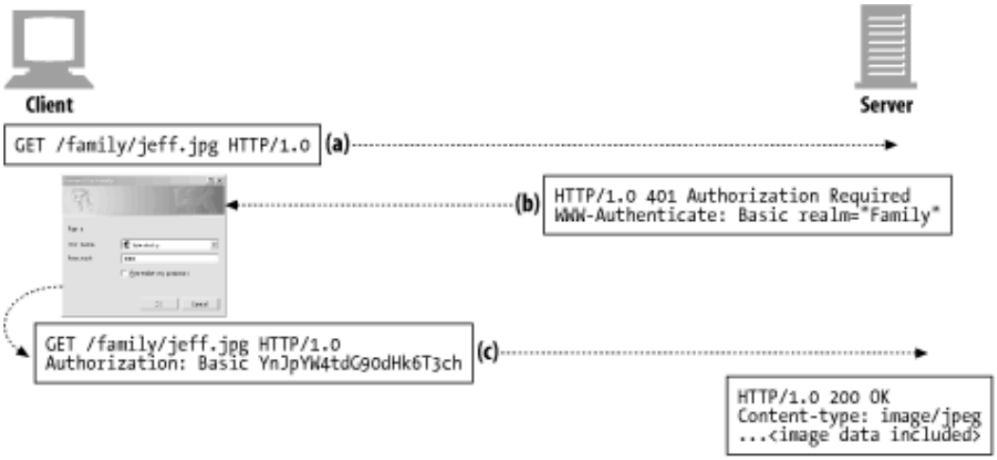
HTTP는 제어 헤더를 통해 다른 인증 프로토콜에 맞춰 확장 가능합니다.

HTTP 인증 헤더에 인증 프로토콜을 기술합니다.

예시로,

단계	헤더	설명	메서드/상태
요청		첫 번째 요청에는 인증 정보가 없다.	GET
인증 요구	WWW-Authenticate	서버는 사용자에게 사용자 이름과 비밀번호를 제공하라는 의미로 401 상태 정보와 함께 요청을 반려한다. 서버에는 각각 다른 비밀번호가 있는 영역들이 있을 것이므로, 서버는 WWW-Authenticate 헤더에 해당 영역을 설명해 놓는다.	401 Unauthorized
인증	Authorization	클라이언트는 요청을 다시 보내는데, 이번에는 인증 알고리즘과 사용자 이름과 비밀번호를 기술한 Authorization 헤더를 함께 보낸다.	GET
성공	Authentication-Info	인증 정보가 정확하면, 서버는 문서와 함께 응답한다. 어떤 인증 알고리즘은 선택적인 헤더인 Authentication-Info에 인증 세션에 관한 추가 정보를 기술해서 응답하기도 한다	200 OK

Figure 12-2. Basic authentication example



1. 서버가 클라이언트에게 **인증요구**

서버는 **401 Unauthorized 응답**과 함께 WWW-Authenticate헤더를 기술해서 어디서 어떻게 인증할지 설명
2. 클라이언트가 서버로 **인증 정보 첨부 및 요청**
   인코딩된 비밀번호와 그 외 인증 파라미터들을 **Authorization 헤더에 담아서 요청**
3. 성공적으로 완료되면 **정상적인 상태 코드(200 ok)를 반환**한다. 추가적인 인증 알고리즘에 대한 정보를 Authentication-Info 헤더에 기술할 수도 있습니다.

보안 영역(Security Realms)

어떻게 각 리소스별로 다른 접근 조건을 다룰 수 있는가?

웹 서버는 **저마다 다른 사용자 권한을 요구**하는 **보안 영역(realm)** 그룹으로 나눕니다.(www-Authenticate헤더의 realm 지시자)

예시로,

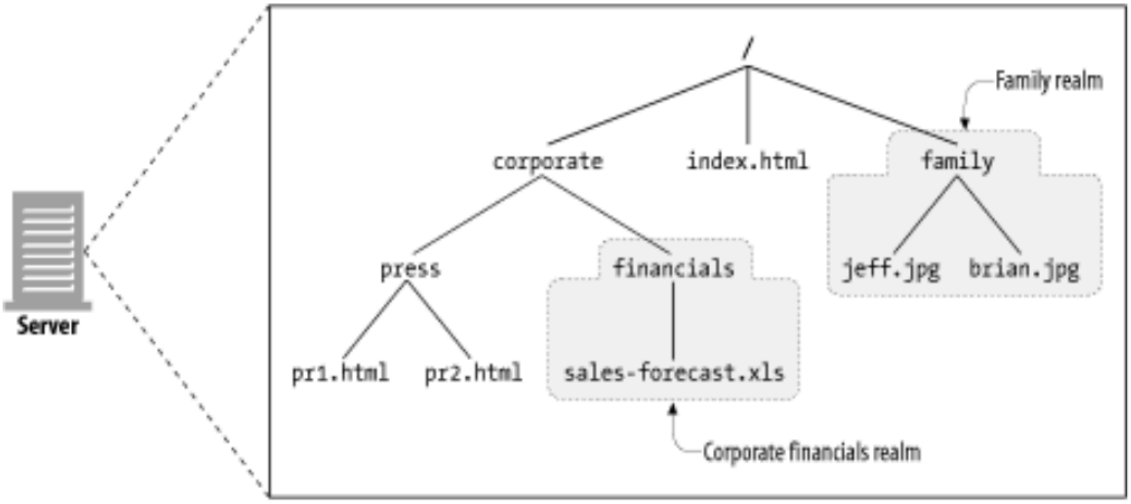
```
HTTP/1.0 401 Unauthorized
WWW-Authenticate: Basic realm="Corporate Financials"
```

위는 realm 파라미터가 기술된 기본 인증입니다.

"Corporate Financials"(회사 재무) 같이 해설 형식으로 되어있어 사용자가 권한의 범위를 이해하는데 도움을 줍니다.

인증이 되면 회사 재무 리소스에는 접근 가능하고, family 리소스에는 별도로 다른 인증이 요구됩니다.

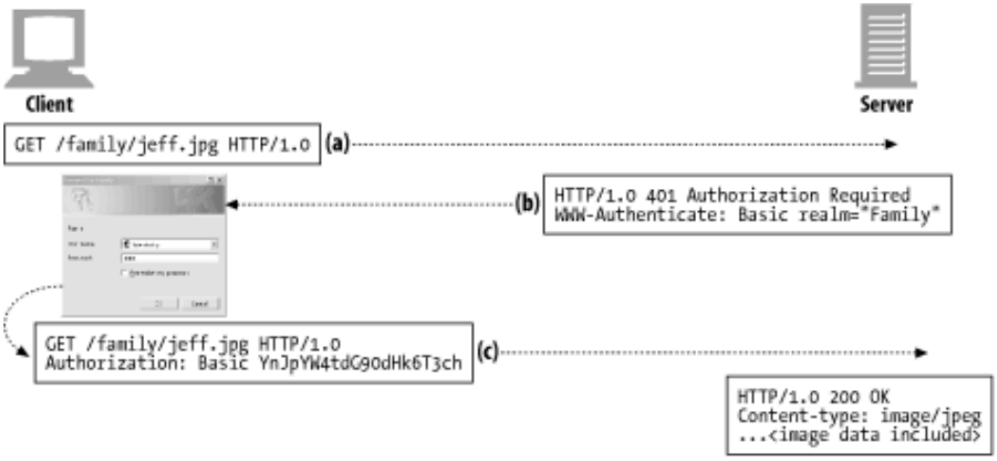
Figure 12-3. Security realms in a web server



기본 인증

거의 모든 주요 클라이언트, 서버에 기본 인증이 구현되어 있습니다.

Figure 12-2. Basic authentication example



위 그림에 기본 인증 절차의 예가 나와있습니다.

1. 사용자가 /family/jeff.jpg **요청**
2. 서버가 **www-Authenticate 헤더(realm 정보)**와 함께 **해당 리소스에 접근하는 데 필요한 인증정보(이름, 비밀번호)를 요구**하는 **401 Authorization Required 응답**을 반환
3. 브라우저에서 사용자는 **인증정보(이름, 비밀번호) 입력**하고, 브라우저는 정보들을 콜론으로 이어 붙이고, **base-64방식 인코딩**을 하여 **Authorization 헤더에 담아 서버**로 다시 요청
4. 서버가 **디코딩하고 값이 정확한지 검사** 후 **200 ok 메세지와 함께 요청 받았던 리소스를 보냄**

> 기본 인증 프로토콜은 Authentication-Info 헤더를 사용하지 않

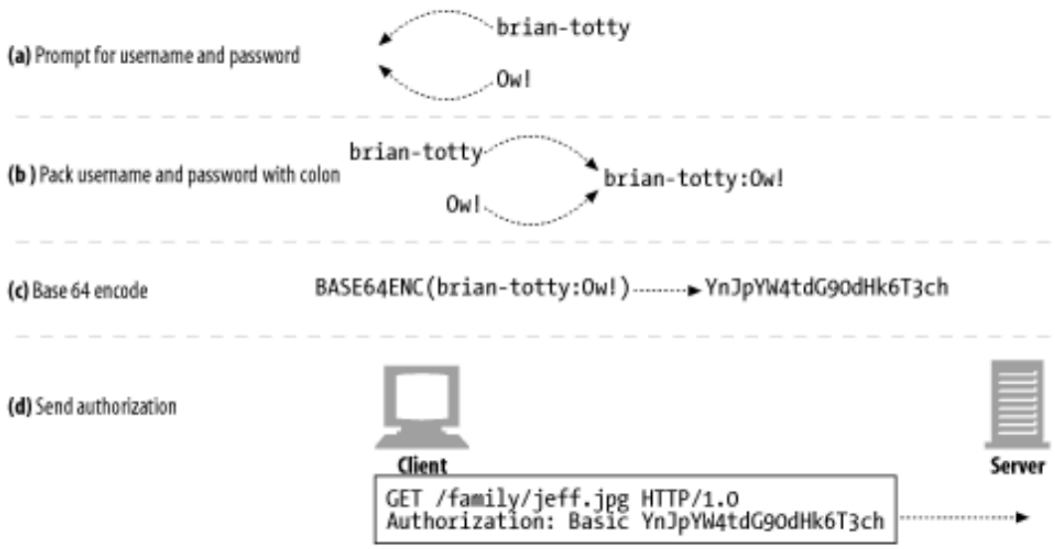
Base-64 사용자 이름/비밀번호 인코딩

> Base-64 인코딩은 바이너리, 텍스트, 국제 문자 데이터(문제를 일으킬 수 있는) 문자열을 받아서 전송할 수 있게, 해당 문자열을 전송 가능한 문자인 알파벳으로 변환하기 위해 발명되었습니다.
>
> 따라서 HTTP 헤더에서 사용할 수 없는 콜론(:), 캐리지 리턴과 라인피드(CRLF), 큰따음표(")를 포함한 사용자 정보를 보낼 때 유용합니다.

8비트로 이루어져 있는 시퀀스를 6비트의 시퀀스로 변환합니다.

각 6비트 조각은 대부분 문자와 숫자로 이루어진 특별한 64개 문자 중에서 선택 됩니다.

Byte character	a (97)								b (98)								c (99)							
8 bit value	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1
6 bit value	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1
6 bit character	Y (24)								W (22)								J (9)							
																				j (35)				



프록시 인증

중개 프록시 서버를 통해 인증할 수도 있습니다.

서버에 접근하기 전에 프록시 서버를 거치게 해서 사용자 인증을 합니다.

-> 접근 정책을 중앙 관리 할 수 있기 때문에 회사 리소스 전체에 대한 **통합적인 접근 제어**가 가능합니다.

프록시 인증은 **웹 서버의 인증과 헤더, 상태 코드만 다르고 절차는 같습니다.**

Web server	Proxy server
Unauthorized status code: 401	Unauthorized status code: 407
WWW-Authenticate	Proxy-Authenticate
Authorization	Proxy-Authorization
Authentication-Info	Proxy-Authentication-Info

기본 인증의 보안 결함

기본 인증은 안심할 수 없기 때문에 의도치 않은 접근을 막는데 사용하거나, ssl같은 암호 기술과 같이 쓰입니다.

기본 인증의 보안 결함으로는,

1. 개인 정보를 쉽게 디코딩할 수 있는 형식으로 네트워크에 전송(base-64는 너무 단순함)
2. 복잡한 방식으로 인코딩되어 있어도, 그 자체를 서버에 보내 인증에 성공하여 접근 가능 - 이에 대한 예방책이 없음
3. 사용자는 같은 다른 서비스들에도 인증정보를 중복해서 사용하는 경우가 많기 때문에 한 곳의 기본 인증 보안이 뚫리면 다른 서비스에 접근 가능
4. 메세지 인증 헤더외에 다른 부분을 수정하여 트랜잭션의 본래 의도를 바꿔버리면(프록시나 중개자가 개입하는 경우), 기본 인증은 정상 동작을 보장하지 않음
5. 가짜 서버의 위장에 취약 - 사용자가 기본 인증을 수행하는 검증된 서버에 연결되어 있다고 믿으면 인증정보를 그대로 넘겨주게 되는 것

즉, 기본 인증은 일반적인 환경에서 개인화나 접근을 제어하는데 편리하고, 다른 사람들이 보지 않기를 원하기는 하지만, 보더라도 치명적이지 않은 경우 유용한 것임.

-> 기본 인증에서 개선된 프로토콜이 다이제스트 인증(13장)

— 주석의 OAuth설명 —

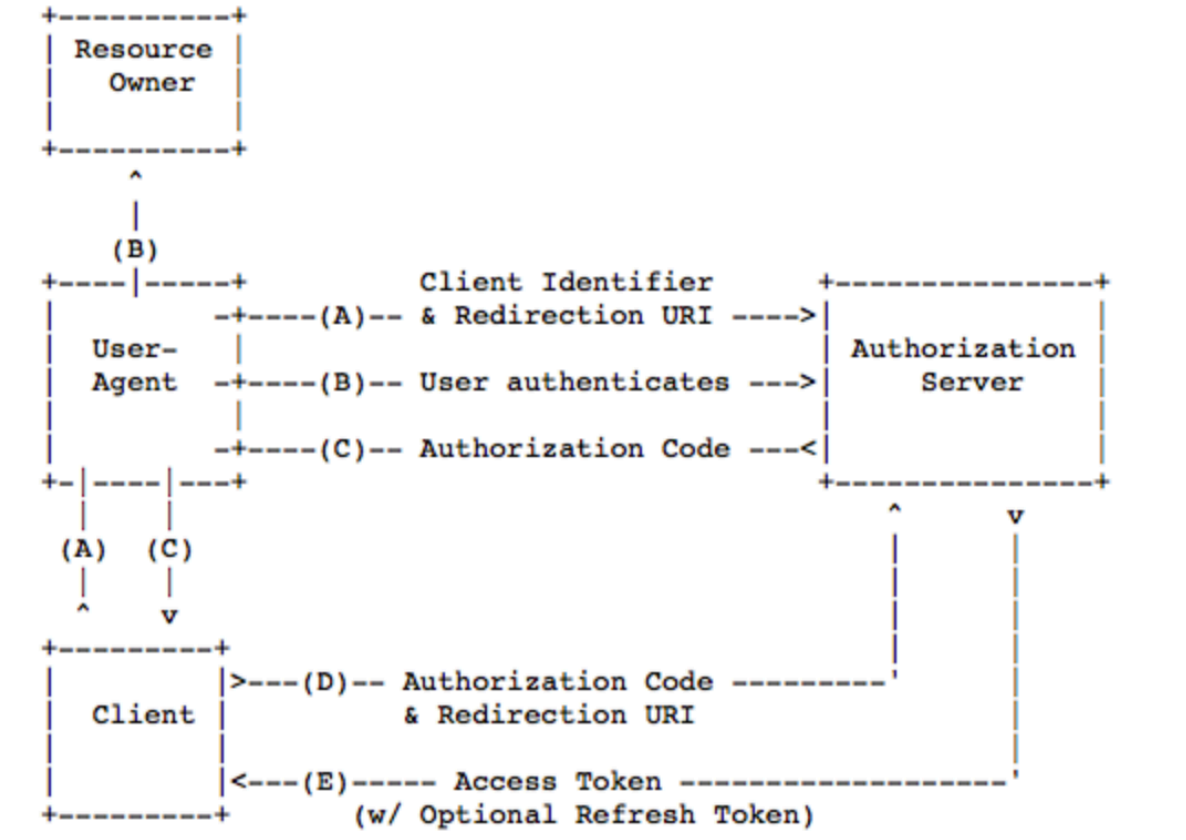
OAuth

Open Authorization, Open Authentication을 뜻하고 네이버, 구글, 트위터(Service Provider)의 **인증 정보를 애플리케이션에 제공 하지 않고 인증, 인가를 할 수 있는 오픈 스탠다드 프로토콜**입니다.

서비스마다 고유의 인증 방식을 사용함에 따라 **인증방식의 표준이 없었고, 보안상 취약한 구조인 기본 인증을 개선하기 위해** OAuth가 나오게 되었습니다.

OAuth의 인증은 API를 제공하는 서버에서 진행하고, 유저가 인증되었다는 Access Token을 발급하는 방식입니다. 이 토큰으로 애플리케이션에서는 Service Provider의 API를 안전하고 쉽게 사용할 수 있게 됩니다.

1.0, 2.0 버전이 있고 여러가지 인증 종류가 있고 그 중 Authorization Code Grant 타입이 아래 그림과 같은 flow를 가집니다.



Note: The lines illustrating steps (A), (B), and (C) are broken into two parts as they pass through the user-agent.

Figure 3: Authorization Code Flow

Client(인증을 사용하는 애플리케이션) / Resource Owner(사용자)

1. Client는 User-Agent를 **인증 서버로 Redirect**(client-id와 redirect-uri 정보도 같이 인증서버로 감)

2. Redirect되면 **사용자의 인증이 시작**되고, 로그인이 안돼있으면 로그인, Application 설치가 안돼있으면 설치한다고 나오면서 권한허가를 구하고, 다 되어있으면 **다시 Client로 Redirect**(1번의 redirect-uri로)
3. 이번에는 **직접 Authorization Server에** client-id, client-secret(App의 비밀번호 역할), 그리고 아까 보낸 code, 그리고 access-token을 받을 uri를 Query String에 담아서 보냅니다.
4. **Authorization Server는 정보가 맞는지 확인** 후, **사용자에게 Access Token을 전송** - 보통 Json
5. 이 후 **Access Token을 사용해 api 이용하여 리소스에 접근 가능**

http://prokuma.kr/oauth/2016/08/04/what-is-oauth.html

https://minwan1.github.io/2018/02/24/2018-02-24-OAuth/

https://d2.naver.com/helloworld/24942

http://interconnection.tistory.com/76