



Onsdag 28 Mars 2018

Databasteknik & PHP

70 YHP

Idag kan vi det mesta 😊

- PHP, input & output
- Funktioner och Scope
- Variabler
- GET vs POST vs PUT vs DELETE
- Klasser och OOP
- Session och Cookies
- Databaser
- SQL DDL & SQL DML
- ER diagram med normalisering

Nu ska vi gå in på det sista sista

- **Övning i Objektorienterad programmering tillsammans databaser**
- **Säkerhet i PHP och SQL**

Övning i OOP med Databaser

- Ladda hem eller forka 'sql-from-php', ni behöver antagligen ändra en del i koden (inloggningsuppgifter, dbnamn mm)
- Kom på ett sätt för att kunna skapa olika **klasser** från olika **tabeller** i databasen
- *Ni kanske inte hinner koda er lösning, men viktigast är att ni kan förklara er idé*



Säkerhet



Cross Site Scripting

Det är ibland möjligt att skicka med javascript till en php fils output

Cross Site Scripting

Det är ibland möjligt att skicka med javascript till en php fils output

```
$search = $_GET['search'] ?? null;  
echo 'Search results for '.$search;
```

Cross Site Scripting

Det är ibland möjligt att skicka med javascript till en php fils output

```
$search = $_GET['search'] ?? null;  
echo 'Search results for '.$search;
```

[http://example.com/search.php?search=<script>alert\('test'\)</script>](http://example.com/search.php?search=<script>alert('test')</script>)

Cross Site Scripting

Det är ibland möjligt att skicka med javascript till en php fils output

```
// This can be solved with htmlspecialchars  
$search = htmlspecialchars($search, ENT_QUOTES,  
'UTF-8');  
echo 'Search results for ' . $search;
```

[http://example.com/search.php?search=<script>alert\('test'\)</script>](http://example.com/search.php?search=<script>alert('test')</script>)

Code injection

I PHP finns ett kommando som heter "eval" som kör php koden som är skriven i en textsträng

Code injection

I PHP finns ett kommando som heter "eval" som kör php koden som är skriven i en textsträng:

```
eval('echo "hello world";');
```

Code injection

I PHP finns ett kommando som heter "eval" som kör php koden som är skriven i en textsträng:

```
eval('echo "hello world";');
```

Kan t.e.x. användas för att hämta kod sparad i databasen 🤯

Code injection

Här måste man också vara försiktig med information som skickas från användaren:

```
eval( 'include ' . $_GET[ 'path' ] );
```

Remote file inclusion

Denna kod kan också vara farlig:

```
$page = $_GET['page'] ?? 'home'
```

```
require $page . '.php';
```

SQL injection

Det vanligaste problemet även med stora hemsidor

Problemet kan sammanfattas till att, i.o.m. SQL queries körs med dynamisk data så kan SQL keywords följa med i den dynamiska datan

...så som att lägga till OR i en where sats, eller ett extra komma i en INSERT.

<https://php.earth/docs/security/sql-injection>

Roligt med SQL Injections:



<https://beta.companieshouse.gov.uk/company/10542519>

SQL injection kan fixas med prepared statements

<http://example.com/get-user.php?id=1> OR id=2;

```
$id = $_GET['id'] ?? null;
```

```
// bump! sql injected code gets inserted here. Be careful to  
avoid such coding
```

```
// and use prepared statements instead
```

```
$query = "SELECT username, email FROM users WHERE id = " . $id;
```

<https://php.earth/docs/security/sql-injection>

SQL injection kan fixas med prepared statements

[http://example.com/get-user.php?id=1 OR id=2;](http://example.com/get-user.php?id=1 OR id=2)

```
$query = "SELECT username, email FROM users WHERE id = ?";
```

```
$stmt = $mysqli->stmt_init();
```

```
if ($stmt->prepare($query)) {  
    $stmt->bind_param("i", $id);  
    $stmt->execute();  
    $result = $stmt->get_result();  
}
```

PREPARED STATEMENTS:

https://www.w3schools.com/php/php_mysql_prepared_statements.asp

<https://php.eearth/docs/security/sql-injection>

Publika filer

Man måste vara försiktig så att inställningsfiler inte går att nå från en klientdator.

- Markera viktiga filer som un-readable i Apache
- *eller*, håll viktiga config filer utanför public_html

eg.

/app

/config och

/public_html

<https://php.earth/docs/security/configuration>

Uppladdning filer

När användaren har möjlighet att ladda upp egna filer är det viktigt att tänka på...

- Begränsa filtypen och byt namn på uppladdade filer
- Kontrollera att filerna inte är skadliga
- Spara filerna på ett sätt som dom inte kan köras från en klientdator

<https://php.earth/docs/security/uploading>

Session hijack

Vilken session som tillhör vilken användare sparas i användarens webbkaka, vid vissa fall kan då en attackerare sno sessionId't från en annan användare och komma åt deras sessions

- Spara inte känslig data i session

PHP konfiguration

En rutinerad hacker kan utvinna mycket information om din sida genom att läsa felmeddelanden och annan information från PHP konfigurationen.

- Slå av `display_errors` (men använd `log_errors`)
- Visa inte php versionen i headers (`expose_php`)
- *Finns fler inställningar som efter behov av applikationen kan stängas av*

★HTTPS★

HTTPS innebär att information som skickas mellan servern och klienten är krypterad på vägen, detta löser många säkerhetsproblem idag då information (så som session id't) inte går att "sno på vägen"

- Installeras olika på olika serverlösningar
- Kräver ett certifikat från en tredjepartsleverantör
- Inte en PHP teknik

<https://php.earth/docs/security/ssl>

Sammanfattning

- Lita inte på input från användaren
- Göm felmeddelanden (bra ur säkerhetsperspektiv och niceness faktorn på sidan)
- "Prepare":a alla SQL anrop
- Använd HTTPS
- Lita inte på input från användaren
- ***Byt standardlösenord mm på MYSQL och andra verktyg***

Vidare läsning

- <https://php.earth/docs/security/intro>
- <https://www.addedbytes.com/blog/writing-secure-php-1>
- https://motherboard.vice.com/en_us/article/aekzez/the-history-of-sql-injection-the-hack-that-will-never-go-away
- <https://arstechnica.com/information-technology/2014/01/how-i-lost-my-50000-twitter-username/>

Tack 🙏