UCloud

HOE VEILIG BEN JIJ

Melany Winklaar, Jordi Samson, Jeroen Hoebert & Aaik Oosters

Inhoudsopgave

Inleiding

Voor het vak security hebben wij de opdracht gekregen om een OWASP top-10 webapplicatie te bouwen. Wij hebben hier ons best voor gedaan en zijn gekomen tot vele verschillende toepassingen om dit project OWASP top-10 valide te krijgen.

Het project is terug te vinden onder de volgende link:

https://github.com/aaikoosters/blok12-security

Wij hebben een UCloud gemaakt. Hierin is het mogelijk dat er foto's ge-upload worden. Deze foto's zijn voor jou en voor jou alleen. Aan ons de kunst om dit af te handelen en op te lossen.

A1:2017-Injection

Er zijn twee verschillende punten waar een injectie kan gebeuren: invoervelden (voor het in loggen en registratie) en het uploaden van afbeelding.

Invoervelden

Achter alle invoervelden hebben we gebruik gemaakt van de methoden='POST'. Daarnaast hebben we bij het uitvoeren van een query bind statements gebruikt. Hieronder is een voorbeeld:

```
$sql = "INSERT INTO users (un, pw) VALUES (?,?)";
$stmt= $conn->prepare($sql);
$stmt->execute([$un, password_hash("$pwtry", PASSWORD_BCRYPT, ['cost' => 12])]);
```

Dat houd in dat alle invoer word gelezen als tekst. Als er een stuk script word geüpload dan word het alsnog als tekst gelezen en zou dat script niks kunnen doen.

Upload afbeelding

Het uploaden van een file word gedaan door input='File'. Daardoor kan de gebruiker alleen een file kiezen van zijn of haar eigen computer. De data word ook niet getoond in de inspect mode van de browser. Daar staat dan gewoon: <input type="file" name="image">.

Als een file word geüpload vanaf hier dan gaat het door een aantal validatie stappen heen.

```
//Is the uploaded file an image?
if( (    strtolower( $name_ext ) == 'jpg' ||
         strtolower( $name_ext ) == 'jpeg' ||
         strtolower( $name_ext ) == 'png'
    ) &&
    ( $mimi == 'image/jpeg' || $mimi == 'image/png' ) &&
    getimagesize( $data ) )
{ //The uploaded file is an image
    // Strip any metadata, by re-encoding image to the tmp image
    if( $mimi == 'image/jpeg' ) {
        $img = imagecreatefromjpeg( $data );
        imagejpeg( $img, $temp_file, 100);
    }else {
        $img = imagecreatefrompng( $data );
        imagepng( $img, $temp_file, 9);
    }
```

Zoals hierboven staat worden de volgende controles word uitgevoerd:

1. De extensie van de file is JPG, JPEG en PNG.
2. Daarna word er aan de file gevraagd of het een afbeelding is ($mimi = $_FILES['image']['type'];).
3. Daarna word er aan de file gevraagd wat de afbeelding afmetingen zijn.
4. Als het daardoor komt, geloven we dat het een image is. Alleen gaan we alsnog de file opnieuw maken. Hierdoor word ook de foute code uit de file gehaald.

Door deze stappen kunnen we zeker zijn dat er geen verkeerde file word geüpload met malware daarin.

A2:2017-Broken Authenticatie

Broken authenticatie houd voornamelijk in dat je inlogsysteem beschermd is tegen te zwakke wachtwoorden en brute force attack. Deze twee punten werd dan verwerkt in onze applicatie om te zorgen dat er toch een beveiligd inlogsysteem ontstaat.

Zwakke wachtwoorden

Om te voorkomen dat 'users' te korte wachtwoorden of gemakkelijk wachtwoorden gebruiken werd er gebruik gemaakt van één van de mogelijkheden, een wachtwoord strenght check.

```php
if(strlen($pwtry) < 8 || !preg_match("#[0-9]+#", $pwtry) || !preg_match("#[a-zA-Z]+#", $pwtry)){
    $_SESSION['fout'] = "wachtwoord te kort";
    header('Location: registererror.php');
}
```

Als het wachtwoord korter is dan acht teken of geen letters bevat of nummers dat krijg de 'user' dan een melding dat het wachtwoord niet goed is dat die een ander moet kiezen. Op deze manier wordt er dan voorkom dat te zwakke wachtwoorden worden gebruik.

Een andere manier is ook om de meest voorkomende wachtwoorden te weigeren tijdens het registreren van een 'user'. Je zoekt dan wat de meest voorkomende wachtwoorden zijn en deze dan ook bij de if-stament plaatsen die dan zorgt dat user een melding krijg dat die een nieuw wachtwoord moet gebruiken. Op deze manier zorg je ook dat de 'users' wachtwoord snel gehackt kan worden. Deze laatste manier hebben wij niet geïmplementeerd in onze applicatie maar als dit in productie gaat zullen wij dit zeker implementeren voor meer veiligheid.

Brute force attacks

Brute force attacks houdt in dat een user verschillende combinaties probeert van gebruikersnamen en wachtwoorden keer op keer totdat die de goede combinatie bereikt. Dit is te voorkomen op twee manieren. De eerste is na een paar mogelijkheden een delay zetten na een aantal keren te hebben geprobeerd. Of de tweede is dan een bijhouden elke keer dat een gebruiksnaam probeert in te loggen en na 5 keren deze 'user' dan een time out te geven van ong. 10 minuten. De tweede manier hebben wij dan toegepast in onze applicatie op deze manier:

```php
if($status == ""){
    // User was not logged in before
    $sql ="UPDATE users SET attempt='1' WHERE un= ?";
    $stmt= $conn->prepare($sql);
    $stmt->execute([$un]);

}else if($status == 5){
    // 5 min time out
    $stort =  strtotime("+5 minutes", time());
    $sql ="UPDATE users SET attempt= 'b-$stort'  WHERE un= ?";
    $stmt= $conn->prepare($sql);
    $stmt->execute([$un]);
}else if(substr($status, 0, 2) == "b-"){
    // Account blocked
    $blockedTime = substr($status, 2);
    if(time() < $blockedTime){
       $block = true;
    }else{
       // remove the block, because the time limit is over
       $sql ="UPDATE users SET attempt= '1' WHERE un= ?";
       $stmt= $conn->prepare($sql);
       $stmt->execute([$un]);
    }
}else if($status < 5){     // If the attempts are less than 5 and not 5
    $sql ="UPDATE users SET attempt= $status + 1 WHERE un= ?";
    $stmt= $conn->prepare($sql);
    $stmt->execute([$un]);
}
```

A3:2017-Sensitive Data Exposure

In de hele applicatie word gebruikt gemaakt van de methode=’Post’. De data word nergens getoond.

Als het inloggen niet lukt, dan word de username niet getoond.

Als iemand inlogt, dan zie je nergens als wie je bent ingelogd.

De authenticatie is gedaan in een aparte file. Die file kan niet worden uitgelezen via een inspect. Die word namelijk veilig verborgen voor de gebruiker. Het programma kan erbij komen maar de gebruiker dus niet.

De hash

De wachtwoorden worden bij de registratie gehashed. Daarvoor gebruiken we BCrypt. Hier volgt de code die we gebruiken voor het hashen van een wachtwoord: password_hash("$pwtry", PASSWORD_BCRYPT, ['cost' => 12]) .

Het resultaat hiervan is:

pw
$2y$12$lbRBJ6BYfA9514ROKuK2fetB/c4w.l.xm4dE1vq4no5...
$2y$12$CUoHHysMZXNtxuilxtn5y.8oDL7IktxWdYdUYnCn58R...
$2y$12$R5h63.mcNfRhU2XJMZIEdujAMo.HuNXJjr8ZyC1T7VZ...
$2y$12$pxApyCA2Z6HybN147OjmU.I0xHE38Zy2dazzY.F7zbI...
$2y$12$Vkif8ztXg3GLFvmAfDKcz.VC91bDAHxrDvm1bFLHNO4...
$2y$12$0x/2smJizWbl4Dpu.NQlq.P7B2FwPCiAxGQtQ7d/7Ik...

Alle wachtwoorden zijn versleuteld naar een hash. In het geval dat de data getoond word zal niemand er iets wijzer van word. Zoals alle wachtwoorden hierboven zijn door de hash anders dan elkaar, maar in praktijk zijn ze allemaal 1234.

A4:2017-XML External Entities (XXE)

Voor dit onderdeel zijn we op het oog valide. We hebben geen XML entities draaien waardoor dit voor nu veilig is. Maar op het moment dat wij wel XML-bestanden hadden gehad dan hadden we het op de volgende manier proberen af te vangen.

Dit kan gecontroleerd worden door verschillende tools. Die kunnen controleren op de boomstructuren binnen de XML lagen. Worden tags naar verwachting ook weer gesloten.

Daarnaast wordt er vanuit PHP een standaard meegeleverd die checkt of dit valide is. Dit is de XMLReader. Deze standaard is waar bekend veilig te gebruiken. We zouden het daarom op die manier toegepast hebben.

Daarnaast wil je XML tijdig valideren omdat het problemen op je websites kan tonen die afgevangen hadden kunnen worden.

A5:2017-Broken Access Control [Merged]

Bij het inloggen krijgt de gebruiker een sessie token, deze sessie token bestaat uit een semi-random string. Dit token staat in de cookies van de gebruiker en wordt uitgelezen op iedere pagina die authenticatie vereist.

Alle andere informatie over een gebruiker en een sessie staat in de database. De server krijgt het token van de gebruiker en als dat de token van een bestaande sessie is haalt de server de rest van de gebruikersdata op. Een sessie verloopt na 5 minuten geen communicatie te hebben met de client.

Doordat er op de token na geen relevante informatie aan de cliënt site opgeslagen wordt, kan er ook niet geknoeid worden met de informatie die de cliënt naar de server stuurt voor bijvoorbeeld het aanpassen van een gebruikers-id, het krijgen van andere bevoegdheden of het verlopen van de sessie.

Omdat alles aan de server kant gebeurd en de cliënt geen verantwoordelijkheden heeft is het erg lastig om hier misbruik van te maken.

Dit is een stukje uit het auth.php file wat overal geïncludeerd wordt.

```
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id, user_id, expiration FROM sessions WHERE sessiontoken = ?");
    $stmt->execute([$oldtoken]);

    // set the resulting array to associative
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    //print_r($stmt->fetchall());
    foreach($stmt->fetchall() as $array)
    {
        if(new DateTime($array["expiration"]) > new DateTime("now"))
        {

            $session_id = $array["id"];
            $user_id = $array["user_id"];
        }
        else
        {
            $logmessage = "expired session";
    include 'super secret logging file.php';
        }
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
    $logmessage = "authenticated";
    include 'super secret logging file.php';
if(!$user_id)
{
    header('Location: secondpage.php');
}
```

Dit is de Sessions tabel:

id	user_id	sessiontoken	time	expiration
1	1	$2y$12$8oq8sjbb4fP7Y02L8BMWXeRt3H4ESdTeDlPo9gvQamR...	2018-12-14 14:59:21	2018-12-14 15:04:21
2	2	$2y$12$JRCtKkrOuEkly4SxJJiMgOkGBv/RwdilznOgM0mxM8u...	2018-12-14 15:00:54	2018-12-14 15:05:54
3	2	$2y$12$fFBvIaF3xKn6aRmobcFfMuYReUzac.edvEIXm00gWME...	2018-12-14 15:00:55	2018-12-14 15:05:55

A6:2017-Security Misconfiguration

Een van de eisen is geen HTTP te gebruiken. Wij wilden dit gaan verhelpen door SSL-certificaten te installeren op onze server. Echter door het gebrek aan een 'live' omgeving hebben wij dit proces overgeslagen. Het voegt in deze situatie niks toe. Het voegt niks toe omdat jij als gebruiker en enige bent die toegang heeft tot jouw omgeving.

Maar wat als we wel online waren gegaan? Dan hadden we dit uiteraard geïmplementeerd. Echter is het ding van SSL-certificaten zelf signed zijn. Dit zorgt voor schijnveiligheid. Het zorgt voor vertrouwen bij de client. Echter is SSL meer het idee van een hek op het land. Waar ja langs kan rijden zonder extra moeite

Scanners

We hebben een scanner gebruikt. Namelijk de OWASP ZAP. De uitkomst was:

Risk Level	Number of Alerts
High	1
Medium	3
Low	4
Informational	0

Er was maar 1 high risk. En dat was dat ze dachten dat er met ' or '1'='1 konden inloggen. Dat is niet mogelijk. Voor meer informatie hierover zie A1. Dit is namelijk een injection.

Daarnaast waren de medium risks' dat je via de browser naar de folder structuur kan gaan. De images zijn namelijk opgeslagen in dezelfde map als de code (of het zit in een map die in dezelfde map als de code staan). Omdat het lokaal is hebben we het zo gedaan. Maar als de applicatie live ging, dan hadden we de afbeeldingen op een andere server opgeslagen.

A7:2017-Cross-Site Scripting (XSS)

XSS is een risico als er input van de gebruiker wordt getoond aan een gebruiker. Aangezien wij dit niet doen is er geen risico. Omdat gebruikers nooit data van elkaar te zien krijgen en zelf geen input kunnen geven achten wij het Cross Site Scripting risico zeer klein.

A8:2017-Insecure Deserialization

Wij slaan geen kritieke data op in cookies en voeren geen gebruikers input uit. Om deze rede gaan we ervan uit niet vatbaar te zijn voor Insecure Deserialization. Echter worden er wel afbeeldingen geüpload, hierbij bestaat natuurlijk het risico dat dit geen afbeeldingen zijn maar kwaadaardige code. Om dat af te vangen hebben we een paar maatregelen genomen:

- De extensie wordt gecontroleerd
- Het bestandstype wordt gecontroleerd
- Het formaat wordt gecontroleerd

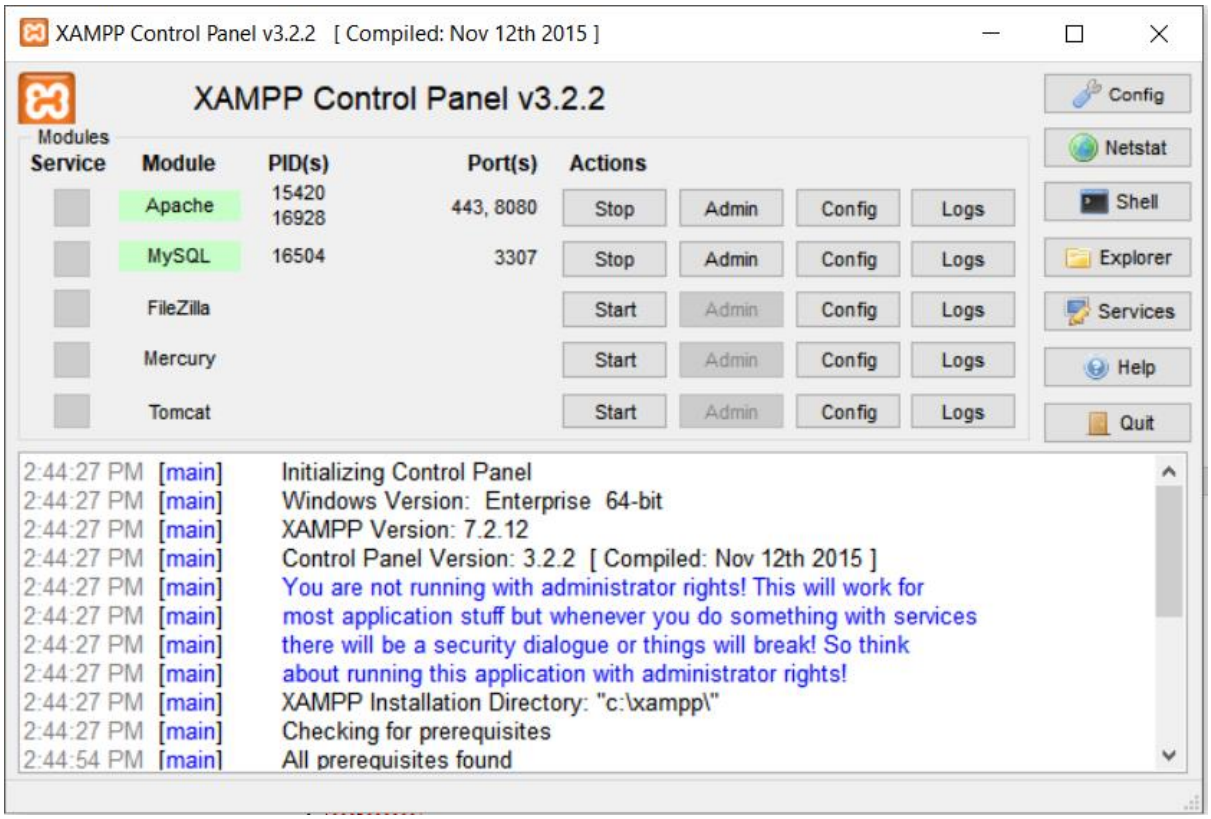
```
//Is the uploaded file an image?
if( (    strtolower( $name_ext ) == 'jpg' ||
         strtolower( $name_ext ) == 'jpeg' ||
         strtolower( $name_ext ) == 'png'
    ) &&
    ( $mimi == 'image/jpeg' || $mimi == 'image/png' ) &&
    getimagesize( $data ) )
```

A9:2017-Using Components with Known Vulnerabilities

Om dit punt toe te passen hebben wij gezorgd dat het programma die we gebruiken van de officiële site komt. We hebben ook ervoor gezorgd dat we gebruik makten van de laatste versies van dit programma omdat dit dan minder kwetsbaarheden bevatten.

We hadden gebruik gemaakt van:

- XAMPP Version: 7.2.12



En de code is geschreven in:

- PHP

A10:2017-Insufficient Logging & Monitoring

Om een applicatie echt veilig te maken is het belangrijk om verdachte activiteiten te kunnen detecteren en hier zo nodig op te reageren. Onze applicatie doet dat door het laden van iedere pagina binnen het portal bij te houden. Ook worden gefaalde login en registreerpogingen bijgehouden en brut force activiteiten geblokkeerd. De logs die worden gegenereerd staan in de SQL-database waardoor ze makkelijk te verwerken zijn zodra hier vraag naar is.

Echter, wat ons systeem niet doet, maar wat wel belangrijk zou zijn om in productie te doen is het monitoren van de communicatie door een extern systeem op het netwerk. En het reageren op overige “verdachte” situaties. Ook zouden de logs die door het systeem worden bij gehouden ergens anders opgeslagen worden zonder de mogelijkheid van het systeem om deze na schrijven aan te passen.

Wel hebben we een “incident response and recovery” plan. Mocht onze applicatie ongewenst gedrag vertonen of onveilig blijken zal er downtime volgen. Dit is de enige manier dat wij kunnen garanderen aan onze gebruikers dat hun gegevens veilig zijn. Echter. Deze downtime zal duren tot het probleem is gevonden en opgelost.

De SQL log ziet er zo uit:

id ▾ 1	session_id	url	content	time
134	25	http://localhost/blok12-security/portal.php	authenticated	2018-12-14 16:37:26
133	25	http://localhost/blok12-security/portal.php	authenticated	2018-12-14 16:36:33
132	*NULL*	http://localhost/blok12-security/loginhandeler.php	loged in, username: 321	2018-12-14 16:36:33
131	*NULL*	http://localhost/blok12-security/loginhandeler.php	login not succesfull, username: 321	2018-12-14 16:36:21