phpBB Security

van

Paul Sohier (0806122)

CMI-Opleiding *Technische Informatica* – Hogeschool Rotterdam

4 oktober 2011

Eerste docent *P.J. den Brok*
Tweede docent

Samenvatting

phpBB is een op het internet bekend geworden software voor het maken van een eigen forum. Het in 2000[11] begonnen project is op dit moment nog steeds actief, maar heeft veel probleme met security gehad. In de afgelopen jaren zijn er grote wijzigingen geweest binnen phpBB om te zorgen dat de security problemen die er zijn geweest voorkomen kunnen worden.

Inhoudsopgave

Inleiding

In dit verslag probeer ik te kijken wat de security problemen zijn van phpBB in het verleden en wat de oorzaak hiervan is. phpBB heeft een aantal jaren geleden intern een groot aantal wijzingen aangebracht in de manier van werken om security problemen te voorkomen. In de afgelopen 3 jaar zijn er binnen de phpBB core geen grote security problemen opgetreden[12].

Niet alleen het core product van phpBB heeft last van security problemen, ook MODifications[1] hadden security problemen. Ook bij deze MODs is geprobeerd om het aantal security problemen wat er was terug te brengen tot een minimaal aantal. In dit verslag besteed ik geen aandacht aan MODs, maar veel MODs hebben dezelfde soort problemen als phpBB zelf.

phpBB had sinds de problematische 2.0 branch een slechte naam op het gebied van security, maar sinds de 3.0 release staat phpBB bekend om zijn goede security en het snelle oplossen van problemen.

[1] MODifications (Ook wel MODs genoemd) zijn aanpassen aan de code van phpBB welke extra functionaliteit toevoegen aan het product

De historie van phpBB

Toen James Atkinson met phpBB begon in 2000 was het bedoeld als “concurent” van UBB, wat al bestaande forum software was. In eerste instantie was phpBB niet heel populair en werd er dus ook weinig aandacht gegegeven aan security. Toen in 2002 phpBB2 uitkwam was phpBB wel al een stuk populairder, maar was er nog steeds weinig gedaan aan security. Dit was helaas ook te zien in de jaren die erna kwamen, doordat ze vol zaten met security releases. Sinds 2.0.4 tot 2.0.23 (Verspreid over 6 jaar) losten alle versies een probleem op gerelateerd aan security. Niet alle problemen waren even groot, maar er waren hierop een aantal uitzonderingen.

Hieronder staat klein overzicht van de problemen welke phpBB heeft gehad in de afgelopen jaren, met de grootste problemen welke er de afgelopen jaren zijn geweest.

Op 14 november 2004 werd phpBB 2.0.11 uitgebracht[6] welke een drietal security problemen oplosten. Deze security problemen waren op 18 november 2004 al gemeld en een oplossing was publiekelijk gemaakt[4] om te zorgen dat zo snel mogelijk mensen zouden updaten. Helaas bleek later dit niet het geval[7]. Doordat het probleem wat gevonden was zo serieus was hadden hackers een worm ontwikkeld welke alle phpBB fora op internet afging en ging kijken of een forum vatbaar was voor het 2.0.11 probleem. Doordat veel fora niet geüpdated waren, kon deze worm een hoop fora hacken. De worm pasten dan alle bestanden welke schrijfbaar waren aan met een tekst dat de site “Defaced” was. Deze worm, genaamd de Never Ever No Sanity worm (Ook wel Santy worm genoemd), kwam “uit” op 20 december 2004[15], ruim een maand nadat phpBB een update had uitgebracht voor de software en gebruikers hadden dus kunnen updaten binnen deze tijd. Om te zorgen dat de Worm minder goed werkten, besloot Google om gebruikers die zochten naar phpBB een foutmelding te geven[2].

phpBB reageerde met de 2.0.11 release goed door snel een patch uit te brengen voor het security probleem wat gevonden was. Het grootste probleem was echter dat gebruikers van de software de software hierna niet bijwerkten naar de nieuwste uitgebrachten release van phpBB. Om dit op te lossen besloot het development team om een aantal verbeteringen hierin aan te brengen zodat gebruikers sneller op de hoogte gesteld konden worden van nieuwe releases. Een goed voorbeeld hiervan was een versie checker in de software, welke controleerde op het forum zelf of de gebruiker de laatste versie had[8].

Het soort probleem wat optrad in phpBB 2.0.11 te voorkomen door gebruik te maken van vooraf gestelde methode om data te verwerken welke door de user is ingevoerd. In phpBB2 was er geen standaard manier om data te verwerken en leverde user input veel problemen op met diverse security issues tot gevolg. Ook in 2.0.13 was dit het geval. phpBB maakt gebruik van cookies om te controleren of een gebruiker ingelogd is of wanneer dit niet het geval is of een gebruiker automatische ingelogd mag worden. In de code voor dit automatische inloggen zat echter een fout waardoor als je het cookie aanpasten welke op je computer stond je automatische kon inloggen, als elke gebruiker. Een hacker kon dus op deze manier ook inloggen als een beheerder van het forum

en op deze manier alles aanpassen wat hij wou. De oplossing voor de probleem was ontzettend eenvoudig, php is standaard niet type strict waardoor een vergelijking zoals hieronder uitvoerd, het eindresultaat true is.

```
1 if (true == 1)
```

Dit betekend alleen wel, wanneer je wel een type stricte vergelijking wilt uitvoeren, je hierbij in je statements rekening mee moet houden. In het geval van de bug inphpBB 2.0.13 moest er een type strict statement uitgevoerd worden, maar dit werd niet gedaan. Dit soort problemen zijn in php een stuk lastiger te vinden zodra ze eenmaal aanwezig zijn. Ook hacker hebben er ruim 2 jaar over gedaan om dit probleem in de source code te vinden. phpBB heeft met de release van 2.0.13 snel gereageerd om het probleem op te lossen, phpBB 2.0.12 was 6 dagen eerder released, maar het probleem was toen nog niet bekend.

Naast de bovenstaande problemen zijn er ook in een aantal versies XSS gevonden en opgelost. Deze problemen kwamen deels doordat het was toegestaan om HTML te gebruiken in berichten op het forum. De HTML werd wel gefilterd, maar er zaten diverse bugs in deze filtering wat voor problemen zorgde.

Naast de diverse problemen in phpBB zelf heeft de site van phpBB zelf ook diverse malen last gehad om zelf gehacked te worden. Door de intern gebruikte software niet up to date te houden is het diverse malen gebeurd dat de complete server van phpBB gehacked was en alle interne fora op torrent netwerken te vinden waren. Ook om dit in het vervolg te voorkomen zijn er intern een aantal verbeteringen doorgevoerd zodat dit niet meer gebeurd.

De verbeteringen

Om het imago te verbeteren van phpBB werd er besloten om in phpBB 2.2 (Welke later phpBB 3.0 werd[9]) de aandacht te leggen op security. In phpBB2 zijn een groot aantal maatregelen genomen om de security te verbeteren zonder inbreuk te doen op de werking van de software. De belangrijkste maatregel hierin was het strict regelen van de manier waarop phpBB geschreven is. Dit is vastgelegd in een document[10] en alle code voor phpBB moet voldoen aan deze richtlijnen. Een belangrijk deel hierbij was dat user input, wat in veel gevallen zorgden voor security problemen, op een vast manier afgehandeld wordt.

Voorkomen van SQL Injection

SQL injection is niet alleen in phpBB, maar in veel websites een groot probleem. Veel ontwikkelaars letten bij de ontwikkeling van een website niet op de security van de website. Dit zorgt ervoor dat veel websites vatbaar zijn voor een groot aantal security problemen.

Bij SQL injection is het mogelijk voor een persoon om queries uit te voeren terwijl dit uiteraard niet toegestaan is. In sommige gevallen betekend dit dat een gebruiker alleen data kan lezen, zoals bijvoorbeeld wachtwoorden, in andere gevallen betekend dit dat een gebruiker ook daadwerkelijk data kan aanpassen welke in de database staan. Het is dus belangrijk om te voorkomen dat er SQL injection aanwezig is in een site.

Een veelgebruikte oplossing tot het voorkomen van SQL Injection is het gebruik van zogenaamde “prepared statements”. Hierbij wordt de variable (Welke in veel gevallen user input is) niet direct in de Query opgenomen, maar wordt op die plek een vraagteken opgenomen. De SQL parser (Bijvoorbeeld PDO of een eigen ontwikkelde parser) zal de data welke opgegeven wordt op dit moment automatische escapen, afhankelijk van het type van de string.

In listing 1 staat een voorbeeld van hoe het duidelijk niet moet. Hier komt de user input direct in de query te staan en kan de gebruiker dus zelf queries uitvoeren welke hij wilt[2]. In listing 2 staat een voorbeeld van hoe dit daadwerkelijk misbruikt kan worden. In dit voorbeeld zal in principe iedere gebruiker worden opgehaald welke in de tabel staat, en afhankelijk van de code hierna bepaald wat er daadwerkelijk gebeurd.

```
1 $input = $_GET['id'];
2 $sql = 'SELECT * FROM users WHERE user = ' . $input;
3 $result = mysql_query($sql);
```

Listing 1: Zonder escaping

[2]Doordat er in php nog een aantal instellingen zijn die bepaalde data escapen, kan je niet altijd daadwerkelijk alles uitvoeren, echter staat deze instelling in de recente versies van php standaard uit[5] en is de functionaliteit deprecated. De reden hiervoor is dat de php ontwikelaars van mening zijn dat het niet hun taak is het werk van de webontwikkelaar te doen. Magic quotes gaf in sommige gevallen ook rare effecten op de waarde die je krijg uit user input.

```
1 http://www.example.org/index.php?id=user OR 1 = 1
```

Listing 2: Voorbeeld van misbruiken van niet gescapede data

Om listing 1 “beveiligd” te krijgen moet de data uit het `$_GET` request escaped worden met bijvoorbeeld `mysql_real_escape_string`.

Een betere manier om data te escapen is het gebruik van prepared statements. In listing 3 en 4 staat een voorbeeld van hoe prepared statements werken met PDO.

```
1 $stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value
      )");
2 $stmt->bindParam(':name', $name);
3 $stmt->bindParam(':value', $value);
```

Listing 3: Prepared statements in PHP[16]

```
1 $stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (?, ?)");
2 $stmt->bindParam(1, $name);
3 $stmt->bindParam(2, $value);
```

Listing 4: Prepared statements in PHP[16]

In listing 3 worden de parameters op naam assigned in de query, terwijl in listing 4 dit op “count” gedaan wordt van het parameter nummer. Het voorbeeld van het gebruiken van de naam is dat wanneer je de query aanpast en een extra veld toevoegd je nog steeds het goede veld assigned. Doordat PDO intern de data escaped zal bij consequent gebruik van prepared statements SQL injection niet mogelijk zijn.

phpBB heeft besloten om geen gebruik te maken van prepared statements. De meest belangrijke reden hiervoor was dat de huidige phpBB versie werd ontwikkeld het gebruik van prepared statements nog niet erg ingeburged was. Een ander probleem was dat PDO niet (goed) werkt onder PHP4 wat nog een minimum eis was van phpBB. Doordat PDO uitgebreid getest en bewezen is kan je daarover zeggen dat het correct werkt en dus veilig is. Wanneer je zelf een SQL parser gaat maken voor prepared statements kan je niet met deze hoge zekerheid zeggen dat het correct werkt.

phpBB heeft om deze reden besloten om gebruik te maken van escaping per specifieke variable en op type. In PHP bestaan er geen types zoals in JAVA waardoor in iedere variable ieder type kan zijn en ook van type kan wijzigen. phpBB heeft intern besloten dat ze dit niet willen en eisen type specifieke variablen voor alles. Hierdoor voorkom je al problemen waardoor een integer verwacht wordt terwijl een string opgegeven wordt en in de query geen single quotes worden gebruikt om aan te geven dat het een string is[3].

Om te zorgen dat alle data type strict wordt afgehandeld is er een simpele interface ontwikkeld om data op te vragen welke user input is (Ofwel, data uit voornamelijk formulieren en data uit de `QUERY_STRING`). Deze interface bestaat uit een simpele methode welke minimaal 2 parameters verwacht. De eerste parameter is de naam van het veld welke opgevraagd moet worden. De tweede parameter is de default waarde welke teruggegeven wordt in het geval de variable niet bestaat in $_REQUEST[4]. Om de type strictness te doen wordt, indien de waarde bestaat, gecast naar het type van de default. In listing 5 staat een voorbeeld van hoe dit werkt. In listing 6 staat de uitkomt van

[3] Uiteraard wordt dit probleem als het goed is al afgevangen door de correct escaping van de data, in het geval van een integer door het aanroepen van `intval` over de variable.

[4] `$_REQUEST` is een samenvoeging van `$_POST` en `$_GET` welke samen alle data bevatten die verstuurd zijn door de gebruiker

deze methode. Het is binnen phpBB niet toegestaan om user input op een andere manier als via `request_var` op te vragen[5].

```
1 var_dump(request_var('test', 1));
2 var_dump(request_var('test', '1'));
3 var_dump(request_var('test', 0.00));
4 var_dump(request_var('test', array()));
```

Listing 5: Voorbeeld van request_var

```
1 int(1)
2 string(1) "1"
3 float(0)
4 array(0) {
5 }
```

Listing 6: Uitkomt van request_var

Het probleem van van SQL injection is hiermee nog steeds niet geheel opgelost. Alle data is op dit moment in principe wel type strict, maar strings moeten nog steeds correct escaped worden om ervoor te zorgen dat er niet uit de single quotes gebruikt kunnen worden. Om deze reden worden in SQL queries strings altijd escaped. Een query zal er in phpBB dus bijvoorbeeld uit kunnen zien zoals in listing 7. Hierbij zie je dat een string in een query altijd in single quotes staat, om aan te geven dat het een string is, terwijl een integer niet in single quotes staat. Doordat de integer al gecast is naar een int, door `request_var` is het niet nodig om dit alsnog te doen, in sommige gevallen bijvoorbeeld wanneer de query in een aparte methode uitgevoerd wordt als waar de data opgehaald wordt, wordt dit alsnog escaped met `intval`. De string is nog niet escaped, dus wordt `$db->sql_escape` aangeroepen welke de data escaped.

```
1 <?php
2 $user = request_var('user', '');
3 $id = request_var('u', 0);
4
5 $sql = 'SELECT user_ID FROM ' . USERS_TABLE . ' WHERE username = \'' . $db->
      sql_escape($user) . '\' OR user_id = ' . $id;
6 $result = $db->sql_query($sql);
7 ?>
```

Listing 7: Query uitvoeren in phpBB

Naast de hierboven genoemde methode heeft phpBB nog een aantal andere methode voor specifieke data, zoals bijvoorbeeld het invoegen van data in een table, welke zelf de escaping afhandelen afhankelijk van het type van de variable.

Een hoop (hobby web ontwikkelaars willen gebruik maken van centraal geregelde escaping welke alle user input die binnenkomt standaard escaped wordt. Hierbij wordt dus ook data welke bijvoorbeeld niet escaped hoeft te worden escaped, of wordt data op de verkeerde manier escaped. Het voordeel is wel dat alle data altijd escaped wordt, en je dus niet kan vergeten dat er data escaped moet worden. Magic quotes in PHP is ook op deze manier bedoeld, dat standaard alle data welke binnenkomt escaped wordt.

[5]Voor phpBB versie 3.0, de huidige stable versie, is dit geregeld via de coding guidelines. Voor de volgende major versie is het zelfs uitgeschakeld om user input op een andere manier op te vragen als via de dan nieuwe request class.

Voorkomen van XSS

XSS is het meest voorkomende security probleem wat er is. XSS staat voor Cross Side Scripting, wat kortweg betekend dat de eindgebruiker HTML kan invoeren op een manier welke niet is toegestaan. Dit wordt vaak veroorzaakt door bijvoorbeeld missende validatie of missende escaping van de HTML.

In eerste instantie lijkt het wel mee te vallen dat je HTML kan invoegen in een document. Maar in veel gevallen kan hier misbruik door gepleegd worden, bijvoorbeeld in combinatie met SQL Injection of CSRF (Cross Site request forgery). Een voorbeeld hiervan is bijvoorbeeld het op de achtergrond door middel van javascript het opvragen van van een externe URL en daarop data uit te voeren.

Een goed voorbeeld van het misbruik van XSS is een probleem in de website van gemeenten, wat digid kwetsbaar maakt[1].

De belangrijkste wijziging ter voorkoming van XSS in phpBB3 is het verbieden van HTML in posts. Bij het gebruik van filtering is de kans op fouten in de filtering zeer groot, mede doordat HTML geen stricte taal is en veel toestaat. Hierdoor is het haast niet te filteren en onstaat bij het gebruik van filtering in veel gevallen problemen, zoals ook in phpBB2 het geval was. Hierdoor mist phpBB wel wat functionaliteit, maar de kans op XSS is hierintegen wel een stuk lager. De data moet uiteraard nog wel escaped worden, anders is XSS via user input alsnog mogelijk.

phpBB heeft hierbij gekozen om alle user data welke binnenkomt via `request_var` en een string is te escapen. De reden dat hiervoor gekozen is is dat bijna alle user input welke binnenkomt daadwerkelijk ook wordt weergeven aan de gebruiker. Op deze manier staat altijd alle data in de database al escaped, en hoeft er bij de weergave niet alsnog gecontroleerd te worden of er nog data escaped moet worden.

password hashing

In phpBB2 werd gebruikt gemaakt van MD5 als hashing mechanisme voor wachtwoord. Op het moment dat phpBB2 ontwikkeld werd was MD5 het defacto standaard voor hashing. In 2007 is echter ondekt dat het MD5 niet langer veilig is door een mogelijke collision[14]. Dit houd kortweg in dat 2 strings (Of bestanden) dezelfde hash hebben, wat in theorie niet mogelijk hoort te zijn. Hierdoor is het gebruik van MD5 afgeraden door diverse instanties en heeft phpBB ook daadwerkelijk besloten om MD5 niet te gebruiken voor password hashing.

phpBB heeft om deze reden besloten om gebruik te maken van phpass[13]. Een belangrijke eis was dat de hash functie altijd moet werken op de door phpBB ondersteunde systemen, welke lopen van PHP 4.3.3 tot PHP 5.3.8 (Op dit moment van schrijven), waarbij ook nog eens bepaalde extensies wel of niet ingeschakeld konden zijn. phpass[3] maakt gebruik van de cryptografische libary van PHP zelf (Welke altijd beschikbaar is sinds PHP 5.3.0), en in het geval deze niet beschikbaar is een eigen implementatie van MD5 welke al sinds PHP3 beschikbaar is. Alle drie de ondersteunde hashing methode maken gebruiken van salt (Het toevoegen van een string welke rainbow tables niet onbruikbaar maken) en herhaalde hashing. phpBB maakt primair gebruik van de md5 fallback, aangezien die altijd beschikbaar is en hierdoor dus portable is naar andere servers[6].

[6] phpBB gebruikt een minimale versie van phpass, waaruit de niet portable hashes, ofwel alle opties welke `crypt` vereisen, niet bevat. De hashes gegenereerd door phpBB zijn compitable met phpass, maar niet alle hashes gemaakt door phpass zijn compitable met de versie van phpass in phpBB.

phpass gebruikt normaal gesproken in eerste instantie, en indien beschikbaar, blowfish als hashing methode. Indien deze niet beschikbaar is wordt er gebruik gemaakt van ext_des, en pas als laatste wordt er gebruik gemaakt van de eigen md5 implementatie.

Voor iedere hash die gemaakt wordt maakt phpass tevens een random salt. Hierbij wordt onder linux en unix gebruik gemaakt van `/dev/urandom`, en indien dit niet beschikbaar is wordt er gebruik gemaakt van een door phpBB eigen geschreven random generator. Deze eigen random generator is gebaseerd op microtime, en is dus niet daadwerkelijk echt random.

De gebruikte hashing methode, aantal herhalingen van de hashing en het salt staan opgeslagen in de string welke wordt teruggegeven. Wanneer een wachtwoord gecontrolleerd moet worden, wordt deze data gebruikt om te controleren of het wachtwoord daadwerkelijk klopt.

Door gebruik te maken van phpass is het wachtwoord op een veilig manier opgeslagen en niet vatbaar voor rainbow tables en brute force op de hash zelf.

phpBB3.1 zal mogelijk als minimale PHP versie PHP 5.3 hebben, waardoor tegen die tijd mogelijk ook de overige hashing methode aan phpBB toegevoegd zullen worden.

Conclusies en aanbevelingen

Sinds phpBB2 is er binnen phpBB al een groot aantal dingen gewijzigd met betrekking tot het denken over veiligheid en zijn er intern richtlijnen gemaakt om security problemen te voorkomen. Hierbij is niet alleen gekeken naar de beste manier om te voorkomen van security issues maar ook naar hoe te handelen wanneer er dan toch nog issues gevonden zijn. Door daadwerkelijk bezig te zijn met security heeft phpBB sinds de release van phpBB 3.0.0 in december 2007 geen enkel groot security issue gehad[12][7]. Dit laat dus zien dat phpBB sinds 3.0.0 eigenlijk goed verbeterd is in vergelijkbaar met phpBB2.

Er valt natuurlijk altijd iets te verbeteren, en dat is ook bij phpBB het geval. In sommige gevallen wordt er nog steeds niet altijd netjes gehouden aan de richtlijnen welke zijn opgesteld. Hierdoor onstaan in sommige gevallen problemen met user input welke eigenlijk voorkomen hadden kunnen worden. Om deze reden zal de volgende major versie dan ook een systeem bevatten welke niet toestaat dat een developer data direct opvraagd maar echt via de interface te werk gaat. Op deze manier wordt er altijd voor gezorgd dat data daadwerkelijk correct afgehandeld wordt. Verder wordt er sinds versie 3.1 ook niet zomaar nieuwe code toegelaten. Voordat code in de development branch wordt toegelaten moet eerst een andere developer buiten de developer die de code geschreven heeft gecontroleerd worden op problemen. Op deze manier wordt er op een vroegtijdig moment voor gezorgd dat er geen security issues worden geintroduceerd in nieuwe code.

[7]Het aantal issues wat secunia meld is niet geheel correct. Bijvoorbeeld SA38837 is geintroduceerd in versie 3.0.7 en enkele dagen daarna al opgelost, secunia zegt echter dat dit in alle versies voor phpBB 3.0.7-pl1 zit. Sommige andere issues zijn door phpBB wel gemarkeerd als security issue, maar zijn eigenlijk meer issues welke problemen voorkomen als dat het issues zelf zijn.

Bronnen

[1] ”blunder logius maakt digid-fraude kinderspel”. http://webwereld.nl/nieuws/108107/lek1--blunder-logius-maakt-digid-fraude-kinderspel.html.

[2] Google stops spread of santy worm. http://www.zdnet.co.uk/news/security-management/2004/12/22/google-stops-spread-of-santy-worm-39181937/.

[3] ”how to manage a php application’s users and passwords”. http://www.openwall.com/articles/PHP-Users-Passwords.

[4] howdark.com exploits - follow up. http://www.phpbb.com/community/viewtopic.php?f=14&t=240513.

[5] ”magic quotes in php”. http://php.net/manual/en/security.magicquotes.php.

[6] phpbb 2.0.11 released - critical update. http://www.phpbb.com/community/viewtopic.php?f=14&t=240636.

[7] phpbb 2.0.11 upgrade reminder. http://www.phpbb.com/community/viewtopic.php?f=14&t=244451.

[8] phpbb 2.0.12 released. http://www.phpbb.com/community/viewtopic.php?f=14&t=265423.

[9] phpbb 2.2 is no more ... meet olympus. http://www.phpbb.com/community/viewtopic.php?f=14&t=256072.

[10] phpbb coding guidelines. http://area51.phpbb.com/docs/30x/coding-guidelines.html.

[11] phpbb history. http://www.phpbb.com/about/history/.

[12] phpbb security problemen. http://secunia.com/advisories/product/17998/?task=statistics.

[13] ”portable php password hashing framework”. http://www.openwall.com/phpass/.

[14] ”predicting the winner of the 2008 us presidential elections using a sony playstation 3”. http://www.win.tue.nl/hashclash/Nostradamus/.

[15] Santy worm makes unwelcome visit. http://news.bbc.co.uk/2/hi/technology/4117711.stm.

[16] ”voorbeeld van prepared statements in php met pdo. http://www.php.net/manual/en/pdo.prepared-statements.php.