RADBOUD UNIVERSITEIT NIJMEGEN

Beveiligingsaspecten van webapplicatie-ontwikkeling

Versie 1.0

12-3-2009

Afstudeernummer :	92 IK
Auteur :	Wouter van Kuipers
Begeleider :	Erik Poll

Inhoud

1 Inleiding

> *The world's six billion people can be divided into two groups: group one, who knows why every good software company ships products with known bugs; and group two, who don't. Those in group 1 tend to forget what life was like before our youthful optimism was spoiled by reality. Sometimes we encounter a person in group two …who is shocked that any software company would ship a product before every last bug is fixed.*
>
> *Eric Sink, Guardian May 25, 2006*

Bovenstaand citaat van Eric Sink is naar mijn mening één van de meest heldere omschrijvingen van het grootste security probleem ooit. Het probleem is namelijk niet dat er securityproblemen zijn maar dat het onmogelijk is om een applicatie te ontwikkelen die nooit last zal hebben van deze problemen. Hoewel dit probleem zich al voordoet sinds het begin van het IT tijdperk is er met de komst van webapplicaties een nieuwe wending gegeven aan het begrip security en is security een belangrijker onderwerp geworden dan ooit te voren.

Naast mijn masterstudie Informatiekunde ben ik als freelance Webdeveloper, dagelijks bezig met de ontwikkeling van webapplicaties in onder andere PHP. Tijdens deze werkzaamheden heb ik vaak te maken met werk dat al door een andere ontwikkelaar is gemaakt of word ik samen met meerdere ontwikkelaars ingezet om samen één applicatie te ontwikkelen. Hierdoor krijg ik vaak te maken met de twee groepen die Eric Sink hierboven beschrijft: de ontwikkelaars die er zeker van willen zijn dat de door hen geprogrammeerde applicatie geen enkel securitybug bevat en aan de andere kant de ontwikkelaars die ervoor zorgen dat de applicatie geen gevaar loopt aangevallen te worden, zonder dat zij hierbij elke security gerelateerde bug opsporen en verhelpen.

Met het onderzoek dat ik voor deze scriptie heb gedaan wil ik vooral de kleine en middelgrote bedrijven in laten zien dat security een belangrijk aspect is van het ontwikkelproces maar dat het niet noodzakelijk hoeft te zijn om alle security gerelateerde problemen uit een webapplicatie te halen voordat deze kan worden geïmplementeerd. Het doel is dan ook om deze bedrijven te helpen de security aspecten van het ontwikkelproces voor webapplicaties te verbeteren.

Dit document is met name bedoeld voor ontwikkelaars binnen bedrijven gericht op webontwikkeling. Een aantal hoofdstukken is ook geschikt voor leidinggevenden, deze raad ik zeker aan om de hoofdstukken 2, 7 en 8 te lezen.

Voor het tot stand komen van deze scriptie wil ik een aantal mensen bedanken: Erik Poll, voor de begeleiding tijdens mijn onderzoek en het geven van nuttige bronnen en tips. Joost Altena , Marc Pieterse en Peter Tenbült voor de ondersteuning tijdens het afnemen en verwerken van de interviews en het controleren van mijn werk. En tot slot mijn ouders en vriendin voor de steun en verbeteren van mijn werk.

2 Management samenvatting

Sinds het begin van deze eeuw zijn webapplicaties een steeds groter wordende trend binnen de IT sector. Applicaties die via het internet beschikbaar worden gesteld aan gebruikers zonder dat hiervoor een programma geïnstalleerd hoeft te worden, krijgen steeds vaker de voorkeur boven de klassieke desktop applicatie. Maar het aanbieden van een programma via het internet brengt vooral op het gebied van security een paar grote gevaren met zich mee, gevaren die desktop applicaties niet ondervonden. Het 'Open Web Application Security Project' (OWASP) heeft in 2007 een top 10 opgesteld met de meest kwetsbare punten van webapplicaties waaronder: niet-gevalideerde invoer van gebruikers, fouten in de toegangscontrole en cross-site scripting. Dit zijn typische problemen die webapplicaties met zich meebrengen doordat de applicaties in de browser worden uitgevoerd en de communicatie over het internet wordt opgebouwd. Dit zijn dan ook de eigenschappen die webapplicaties onderscheiden van de desktop applicaties.

Om er achter te komen hoe het gesteld is met de security aspecten van webapplicatie ontwikkeling heb ik een aantal onderzoeken uitgevoerd. Om de huidige stand van zaken op securitygebied voor webontwikkeling in kaart te brengen heb ik een literatuuronderzoek gedaan, daarnaast heb ik een aantal ontwikkelaars bij verschillende bedrijven geïnterviewd. Vervolgens heb ik de 'Sourcecode analyse Tool' Fortify onderzocht om te kijken hoe een dergelijke tool ontwikkelaars kan ondersteunen bij het analyseren van hun broncode.

Uit mijn onderzoek is gebleken dat bij kleine en middelgrote bedrijven vaak een budget ontbreekt voor het volgen van cursussen en trainingen waardoor de ontwikkelaars zelf verantwoordelijk zijn voor het up-to-date houden van hun kennis. Hierdoor zullen de ontwikkelaars minder literatuur aan schaffen en maken ze gebruik van een gratis alternatief, namelijk het internet. Een bedrijf kan hier goed op in spelen door zelf actief aan kennismanagement te doen, door bijvoorbeeld het opzetten van een Wiki. Hierdoor wordt de communicatie tussen ontwikkelaars niet alleen bewaard voor andere ontwikkelaars maar kunnen ontwikkelaars ook elkaars bevindingen gebruiken, onderbouwen of aanvullen.

Daarnaast is gebleken dat er bij veel ontwikkelbedrijven nog vaak een 'security wordt pas een probleem als het een probleem is' cultuur heerst. Hiermee wordt bedoeld dat er bij bedrijven pas voldoende middelen beschikbaar worden gesteld voor het bestrijden van securityproblemen nadat ze geconfronteerd worden met deze problemen. Dit is een serieus probleem, niet alleen omdat een goed security beleid een actieve aanpak vereist, maar ook omdat blijkt dat ontwikkelaars in de praktijk de securityproblemen erkennen maar onvoldoende middelen krijgen om er daadwerkelijk iets aan te kunnen doen.

Een van die middelen die de ontwikkelaar bijvoorbeeld kan ondersteunen bij het controleren van zijn fouten tijdens het programmeren is een zogenaamde "Sourcecode Analyzer", een programma dat broncode analyseert en onderzoekt op mogelijke security gerelateerde fouten. Uit mijn onderzoek bleek dat een tool als deze de ontwikkelaar niet alleen kan ondersteunen door de code op fouten te controleren maar ook door de ontwikkelaar te leren wat hij fout heeft gedaan en hoe hij ervoor kan zorgen dat hij dit de volgende keer foutloos kan doen. Dit doet het programma door bij elke fout een uitgebreide uitleg te geven over het ontstaan van het probleem, de inhoud van het probleem en de eventuele oplossingen die de ontwikkelaar kan nemen om het probleem op te lossen en te voorkomen dat een zelfde fout in de toekomst nog een keer gemaakt zal worden.

Uit het onderzoek naar Fortify bleek dat deze tool in staat is om broncode snel en effectief te scannen op mogelijke problemen. Gebleken is dat hoewel het programma niet altijd alle fouten zal vinden het scannen van de broncode snel en relatief eenvoudig kan worden uitgevoerd. Wel is gebleken dat het verstandig is om een ontwikkelaar vooral zijn eigen broncode te laten analyseren. Hierdoor zal de ontwikkelaar de gevonden problemen makkelijker kunnen oplossen omdat hij zelf de code waarin het probleem zich voordoet heeft geschreven. Het toepassen van een applicatie als Fortify is dan ook zeker een aanrader ter aanvulling van bestaande middelen voor het analyseren van broncode.

Om ervoor te zorgen dat de ontwikkelaars de middelen krijgen die ze nodig hebben om de securityproblemen aan te kunnen pakken is het verstandig om te werken met een securitybudget. Ook al is er binnen het bedrijf op dit moment geen vast budget voor security doeleinde, dan is het alsnog verstandig een schatting te maken van de hoeveelheid security gerelateerde kosten die zullen worden gemaakt binnen een project. Ik wil benadrukken dat securityproblemen altijd zullen blijven bestaan, zij het in een andere vorm, zij het vanuit andere reden, er zal altijd rekening moeten worden gehouden met het feit dat er aanvallen kunnen plaatsvinden doormiddel van (misschien nu nog onbekende) exploits in een applicatie. Zorg er dus voor dat het security beleid binnen uw onderneming niet alleen is gericht op het huidige security beeld, maar houd rekening met nieuwe ontwikkelingen in de toekomst en zorg er voor dat er bij nieuwe calamiteiten snel en adequaat kan worden ingegrepen.

3 Literatuurstudie

Tijdens de voorbereidingen van het schrijven van dit document heb ik een aantal bronnen geraadpleegd om mijn kennis op het gebied van beveiliging van webapplicaties te vergroten. In dit hoofdstuk zullen de bronnen die ik geraadpleegd heb genoemd worden en zal ik aan geven wat ik goed en minder goed vond aan deze bronnen.

De bronnen die ik geraadpleegd heb zijn onderverdeeld in drie categorieën: bronnen die het ontwikkelproces beschrijven, bronnen specifiek voor PHP ontwikkelaars en bronnen betreffende security tools. Managers die de boeken willen lezen raad ik aan om boeken uit sectie 3.1 'Het ontwikkelproces' of het artikel over 'Web Applications Security Certificates' te lezen, de boeken uit sectie 3.2 'Ontwikkelen met PHP' zijn aan te bevelen voor PHP-ontwikkelaars.

3.1 Het ontwikkelproces

Het ontwikkelproces is de basis waarin security als eerste kan worden toegepast. Als er tijdens het ontwikkelproces al rekening wordt gehouden met mogelijke securityproblemen, zal dit later een hoop tijd en geld besparen. Juist om deze reden heb ik een aantal boeken over het ontwikkelproces van applicaties bestudeerd. Deze boeken zijn niet allemaal specifiek gericht op webontwikkelling. De reden is dat er een overlapping is tussen de maatregelen die genomen kunnen worden bij zowel desktop- als webapplicaties.

Software security – Building Security in [1]

Dit boek richt zich vooral op de veiligheid van applicaties tijdens de ontwikkelfase. Het boek is vooral geschreven voor software-experts, maar ook ontwikkelaars, architecten en technische managers kunnen baat hebben bij dit boek omdat delen ervan heel begrijpelijk voor leken zijn geschreven zonder dat er een grote kennis op het gebied van beveiliging nodig is. In het boek worden steeds kleine voorbeelden uitgewerkt waarbij praktijkvoorbeelden worden gecombineerd met goed gestructureerde oplossingen.

Web Engineering [2]

'Web Engineering' is een boek gericht op het ontwikkelproces van webapplicaties, ongeacht welke programmeertaal of omgeving hiervoor gebruikt wordt. De rode draad in dit boek is de casus van SafeHomeAssured.com waarmee de lezer als het ware betrokken wordt in het ontwikkelproces van deze webapplicatie. Door deze insteek worden alle stadia van het ontwikkelproces belicht en worden problemen en gevaren behandeld die tijdens de verschillende stadia zouden kunnen voorkomen. Een minder goed punt van dit boek is dat alles via een bepaalde ontwikkelmethode wordt doorlopen en dat er geen voorbeelden zijn van hoe het ook anders zou kunnen.

The security development lifecycle [3]

Microsoft staat of liever gezegd stond bij computergebruikers niet bepaald bekend als een bedrijf dat veel aandacht besteedde aan softwarebeveiliging. In dit boek wordt beschreven wat Microsoft de afgelopen jaren heeft gedaan om dit beeld te veranderen en hoe zij beveiliging als een van de belangrijkste factoren binnen hun ontwikkelproces in hebben gebouwd. Tijdens het lezen van het boek krijg je een groot aantal voorbeelden van problemen voorgeschoteld waar de ontwikkelaars van Microsoft tegenaan liepen tijdens en na het ontwikkelen van hun besturingssystemen. Wat hierbij vooral opvalt, is dat hoewel het boek geschreven is door medewerkers van Microsoft er toch veel voorbeelden in staan die niet zozeer Microsoft gerelateerd zijn. Een ander opvallend punt is dat er geen moment wordt geprobeerd om Microsoft in te dekken voor hun eerdere werkwijze op het gebied van beveiliging. Ook de oplossingen die geboden worden, het inbouwen van beveiliging in het ontwikkelproces, zijn inzetbaar in allerlei vormen van softwareontwikkeling, met name op het gebied van webapplicaties.

Development life cycle issues [4]

In maart 2008 heeft Kenneth van Wyck, voor een conferentie georganiseerd door 'The Open Web Application Security Project' (OWASP), een bijdrage gemaakt met als titel `Secure Development Processes`. Deze presentatie ging over de security aspecten van 'development life cycles', waarbij de nadruk op de problemen tijdens deze cyclus werd gelegd. Tijdens deze presentatie maakt Kenneth duidelijk waarom het schrijven van veilige software lastig is en hoe een goed ontwikkelproces dit kan vergemakkelijken.

Als voorbeelden voor goede processtructuren noemt Kenneth de Microsoft 'Secure Development Lifecycle', Citigal's 'Touchpoint process' en het OAWSP 'Comprehensive Lightweight Application Security Process'. Hij vergelijkt deze drie structuren aan de hand van de goede en slechte eigenschappen en geeft aan waar bedrijven op moeten letten bij het kiezen van de juiste structuur.

Deze presentatie is een goed oriëntatiepunt voor bedrijven die geen vast proces hebben voor de ontwikkeling van nieuwe applicaties, maar die dit wel willen gaan vormen in de toekomst. De presentatie laat het nut van een gestroomlijnd proces zien en vergelijkt de verschillende mogelijkheden zodat er een duidelijk beeld ontstaat van welke processtructuur het beste past bij de situatie van het bedrijf.

The Open Web Application Security Project (OWASP) [5]

OWASP is een wereldwijde open community gefocust op het verbeteren van de security van software. Dit proberen ze te bereiken door security op de kaart te zetten zodat organisaties zich bewust worden van security en inzien waarom security voor hun applicaties belangrijk is. Nederland heeft zijn eigen divisie die een aantal keer per jaar bijeenkomsten organiseren waarbij security gerelateerde onderwerpen aan bod komen. Daarnaast is op de website van OWASP veel informatie terug te vinden, waaronder de OWASP top 10. Deze top 10 bevat de 10 meest voorkomende security gevaren en is opgesteld in 2004 en herzien in 2007.

3.2 Ontwikkelen in PHP

PHP is een van de meest gebruikte programmeertalen voor het ontwikkelen van webapplicaties. Recent onderzoek van 'Security Space'[1] wees uit dat PHP nog steeds de meest gebruikte taal is voor het ontwikkelen van webapplicaties. Ikzelf werk al ruim 7 jaar met de taal PHP en in de loop van de jaren heb ik mij verdiept in de securityproblemen die zich specifiek binnen deze taal voordoen. Bij de ontwikkeling van PHP zijn in het verleden een groot aantal concessies gedaan die de security aspecten van de taal niet ten goede zijn gekomen. Deze concessies zijn gedaan om ontwikkelaars de gelegenheid te geven om snel met PHP te kunnen werken en bekend te raken met de taal, iets wat volgens de makers achteraf niet de beste keuze bleek te zijn [6]. Om deze reden zijn er voor PHP dan ook de nodige boeken verschenen met security als onderwerp. In deze sectie zal ik een aantal van de door mij gelezen boeken voorzien van een korte beschrijving.

Pro PHP security [7]

`Pro PHP security` is een PHP-security boek dat zich focust op het hele ontwikkeltraject van een PHP applicatie. Hiervoor is het boek opgedeeld in vier delen waarin respectievelijk wordt behandeld wat het belang is van security, hoe een veilige omgeving gecreëerd en onderhouden kan worden, hoe er veilig in PHP geprogrammeerd dient te worden en tot slot gaat deel vier in op beveiliging van applicaties. Het derde deel van het boek, waarbij het gaat over het veilig programmeren vond ik niet al te sterk. Hiervoor kan beter het boek 'PHP|architect's Guide to PHP Security' [6] voor worden gebruikt omdat deze naar mijn mening de problemen en 'best practices' beter gebruikt en uitlegt.

PHP|architect's Guide to PHP Security [6]

De auteur van dit boek, Ilia Alshanetsky, is zelf één van de ontwikkelaars van PHP. Hierdoor weet hij niet alleen heel veel van de werking van deze programmeertaal, maar kan hij ook de keuzes van het ontwikkelteam onderbouwen. Dit is vooral interessant als het over beveiliging gaat, aangezien er bij de start van het PHP project nogal wat opvallende keuzes gemaakt zijn op dit gebied. Alshanetsky geeft daarnaast een groot aantal 'best practices' voor ontwikkelaars, die niet alleen gelden voor ontwikkelen met PHP maar ook voor andere (web)programmeertalen van toepassing zijn.

Essential PHP Security [8]

Dit boek, vooral gericht op programmeurs die nog minder bekend zijn met PHP, geeft een goede houvast voor het werken met beveiliging in PHP. Met behulp van goede voorbeelden worden mogelijke beveiligingsproblemen van webapplicaties uitgelicht en worden oplossingen aangereikt. De diverse beveiligingsonderwerpen zijn onderverdeeld in de hoofdstukken; formulieren en URL's, databases en SQL, sessies en cookies, includes, bestanden en commando's, authenticatie en autorisatie en gedeelde hosting. De specifieke probleemgevallen die worden behandeld zijn veelal alleen van toepassing voor PHP.

[1] http://www.securityspace.com/sspace/index.html

PHP.NET [9]

De officiële website van PHP is een grote bron van informatie over deze programmeertaal. Zo is op deze website de complete referentie naar alle functies te vinden evenals de complete documentatie en gebruikershandleiding, waarin onder andere een aantal hoofdstukken gerelateerd zijn aan security. Daarnaast bevat de website een subsite genaamd talks.php.net, op deze website staan een aantal presentaties van onder andere PHP ontwikkelaars waaronder een sectie over security.

3.3 Security tools

In deze sectie beschrijf ik de 'Security tools' gerelateerde bronnen die ik heb bestudeerd voor mijn onderzoek. 'Security tools' zijn hulpmiddelen die ontwikkelaars van (web) applicaties kunnen gebruiken voor het testen en/of verbeteren van hun producten op het gebied van onder andere security. Hierbij kan gedacht worden aan programma's die de code van de ontwikkelaar controleren op fouten, maar ook aan informatieve tools die de gebruiker ondersteunen tijdens het ontwikkelproces door informatie of code aan te bieden die de ontwikkelaar kan gebruiken. Naast de onderstaande scriptie heb ik onder andere ook een artikel bestudeerd van Ware & Fox getiteld 'Securing Java Code: Heuristics and an Evaluation of Static Analysis Tools' [10] en een artikel van Ayewah & Puhg getiteld 'A Report on a Survey and Study of Static analysis Users' [11]

Web Applications Security Certificates [12]

Emile Strijbos heeft in oktober 2008 zijn scriptie over 'Web Application Security Certificates' gepubliceerd. Voor zijn scriptie heeft Emile onderzoek gedaan naar de manier waarop security certificaten voor webapplicaties kunnen worden vastgesteld en uitgereikt. Onderdeel van dit onderzoek was onder andere een analyse van de meest voorkomende beveiligingslekken in webapplicaties. Aan bod komen onder andere hoe deze lekken te voorkomen zijn, hoe ze te detecteren en tot slot hoe deze detectie geautomatiseerd kan worden met speciale 'source analysis tools' en 'vulnerability scanners'. De resultaten van zijn onderzoek vormen een belangrijke bron voor mijn eigen onderzoek en deze scriptie.

3.4 Conclusie

Op het gebied van security en (web)ontwikkeling zijn een groot aantal bronnen te vinden. Om tot deze lijst met bronnen te komen heb ik mij laten informeren door collega's, professoren van Radboud Universiteit Nijmegen en een aantal bronnen op het internet zoals Jeff Atwood[2], OWASP en PHP.net.

Bij mijn zoektocht naar bronnen ben ik echter wel tot de conclusie gekomen dat veel materiaal dat gericht is op het verbeteren van de security aspecten van het ontwikkelproces vooral gericht is op grote bedrijven. Dit had ik op voorhand al wel verwacht, maar dit is toch een probleem aangezien vrijwel de meeste ontwikkelbedrijven klein tot middelgroot van omvang zijn [13]. Daarnaast zijn er wel veel security gerelateerde bronnen te vinden over ontwikkeling in het algemeen en webontwikkeling in het bijzonder, vooral over PHP zijn er veel bronnen te vinden.

De kwaliteit van de bronnen was over het algemeen goed, ook de boeken voor grote ondernemingen zijn toch interessant om te lezen voor ontwikkelaars/managers in het midden en kleinbedrijf, aangezien er sommige informatie in staat die ook in bedrijven met een kleinere omvang toepasbaar is.

[2] http://www.codinghorror.com

4 Achtergrond

In dit hoofdstuk ga ik in op de achtergrond van het ontwikkelen van webapplicaties. Hiermee probeer ik inzicht te geven in de problemen die webdevelopment met zich meebrengen, problemen die eerder bij klassieke desktop applicaties niet bestonden of geen bedreiging vormden. Daarnaast geef ik inzicht in de historie van het webdevelopment en kijk ik naar de hedendaagse implementatie van security en de ontwikkelingen voor de toekomst.

Sinds het ontstaan van computer netwerken in de jaren 60 [14], en later het ontstaan van het internet, is het samenwerken tussen verschillende gekoppelde computersystemen niet meer weg te denken uit de computerwereld. Vooral in het bedrijfsleven zijn netwerken onmisbaar en worden zij ingezet om data en programma's te delen binnen het bedrijf.

Deze applicaties op een bedrijfsnetwerk werden in het verleden veelal alleen gebruikt in een afgesloten omgeving, waardoor misbruik en/of diefstal van data gemakkelijk voorkomen kon worden. In sommige gevallen gebeurde dit door het bedrijfsnetwerk fysiek te scheiden van het internet, door het netwerk te beveiligen doormiddel van software- of hardwarematige technieken. Op deze manier konden gebruikers binnen het netwerk beschikken over applicaties en data waar ze ook waren, zolang de software maar geïnstalleerd was op de computer en er een verbinding was met het netwerk.

Doordat de applicaties alleen gebruikt werden binnen het interne netwerk was de beveiliging gemakkelijk te beheren. Juist doordat dit netwerk niet toegankelijk was voor vreemden waren de security risico's klein en beheersbaar. De gevaren kwamen voornamelijk van binnen de organisatie en aanvallen van buitenaf, gericht op specifieke applicaties, waren zeldzaam.

4.1 Webapplicaties versus Desktopapplicaties

Met de komst van webapplicaties was het echter mogelijk om applicaties aan te bieden zonder dat een gebruiker afhankelijk is van een bepaalde locatie of werkomgeving. Een desktopapplicatie die voorheen alleen beschikbaar was op een beperkte locatie, wordt nu vervangen door een web applicatie die wereldwijd beschikbaar is op elk systeem dat beschikt over een browser en een internetverbinding.

4.1.1 De geschiedenis van webapplicaties

Het concept voor een webapplicatie bestaat al sinds de eerste vormen van het ARPA-netwerk [14] maar was pas realiseerbaar sinds de opkomst van JavaScript in 1995, waarmee op de cliënt bewerkingen konden worden uitgevoerd. Voorheen was het alleen mogelijk om op de server bewerkingen uit te voeren, waardoor het werken met een webapplicatie een omslachtig en veelal traag proces was [15].

In 1997 werd door een initiatief van een aantal programmeurs de programmeertaal PHP ontwikkeld. Deze taal was volgens de makers zeer geschikt om websites mee te maken, vooral door niet-programmeurs. Om dit te bereiken hebben de ontwikkelaars ervoor gekozen om de taal laagdrempelig te maken door de leercurve voor deze taal vlak te houden, mede door de beveiligingsaspecten van PHP af te zwakken naar een laag niveau. Op dit moment is PHP de meest gebruikte programmeertaal voor het web met een marktaandeel van 33%[3].

Rond 2005 werd het begrip AJAX geïntroduceerd door onder andere Jesse James Garrett [16] waarmee de kloof tussen desktop- en webapplicaties verder werd verkleind. Door de introductie van dit begrip werd ook het 'buzzwoord' Web 2.0 geïntroduceerd [17], een term waarmee onder andere aangegeven werd dat het internet volwassen begon te worden en nu zelfs gezien kon worden als platform [18].

4.2 Risico's

Door deze ontwikkelingen en de hype rond Web 2.0 raakten veel bedrijven geïnteresseerd in het ontwikkelen van webapplicaties. Voornamelijk de bereikbaarheid, veelzijdigheid en lage systeem- en software-eisen waren voor veel bedrijven een goede reden om applicaties die voorheen een klassieke client-server omgeving gebruikten, om te bouwen naar een webapplicatie of om applicaties te ontwikkelen die voorheen gewoonweg niet mogelijk waren.

De ontwikkeling van webapplicaties zorgde voor een hele reeks aan beveiligingsproblemen die zich nooit voordeden bij desktopapplicaties. Een deel van deze problemen zijn ontstaan door de populariteit van de programmeertalen die op het internet gebruikt worden zoals PHP. Deze talen zijn gemakkelijk te leren en te gebruiken voor het maken van een website. Echter voor het schrijven van een webapplicatie is een hogere kennis van deze talen nodig, vooral op het gebied van beveiliging.

[3] http://www.nexen.net/chiffres_cles/phpversion/18780-php_statistics_for_august_2008.php

4.3 Verschuiven van beveiligingsaspecten

Met de verschuiving van klassieke client-server applicaties naar webapplicaties, verschoven ook de aandachtspunten ten behoeve van beveiligingsaspecten. Dit is een van de redenen waarom het ontwikkelen van een webapplicatie een andere aanpak vereist dan het ontwikkelen van klassieke software.

Daarnaast moet er rekening gehouden worden met de gevaren van het plaatsen van applicaties buiten het eigen (lokale) netwerk, zoals het internet. Zoals eerder aangegeven zitten er aan het aanbieden van webapplicaties meer haken en ogen op het gebied van security dan klassieke software, iets waarbij ook in het ontwikkelproces rekening gehouden dient te worden.

Vooral tijdens de eerste generatie van webapplicaties besteedde ontwikkelaars (te) weinig tijd aan de security van hun applicaties. Het gevolg hiervan was dat grote webapplicaties gemakkelijk aan te vallen waren en security bugs aan de orde van de dag waren [19] [20]. Volgens analisten had dit vooral te maken met de onervarenheid van de ontwikkelaars, terwijl deze juist de schuld gaven aan de managers die te weinig budget vrijmaakte voor de security van de applicaties [21].

4.4 Security in het heden en de toekomst

Inmiddels zijn problemen met security aan de orde van de dag. Elke dag worden er weer kleine en grote beveiligingslekken gevonden in allerhande (web)applicaties. Hoewel er allerlei stappen worden ondernomen door de ontwikkelaars van webprogrammeertalen en browserproducenten blijven de problemen over security bestaan. Er worden nog steeds nieuwe aanvalstechnieken gevonden, wie had er tien jaar geleden gehoord over Cross-site scripting (XSS)? Wie weet welke gevaren webapplicaties over een jaar lopen?

Toch zijn dit zaken waar ontwikkelaars rekening mee moeten houden. In dit document zal ik aan de hand van een casus het ontwikkelproces van een webapplicatie nagaan, vanuit het security perspectief analyseren en, waar mogelijk, verbeteren. Hiervoor ga ik niet alleen de werkwijze na, maar test ik ook software speciaal geschreven voor het analyseren en testen van webapplicaties met het oog op security.

Dit ga ik doen door middel van twee onderzoeken te weten een aantal interviews met ontwikkelaars bij bedrijven die webapplicaties ontwikkelen en door het onderzoeken van een security tool. Met behulp van de interviews met de ontwikkelaars wil ik nagaan wat er op dit moment bij de ontwikkelbedrijven gedaan wordt om securityproblemen tegen te gaan en op te lossen. Daarnaast wil ik nagaan wat er binnen het ontwikkelproces wordt gedaan op het gebied van security en of er gebruik gemaakt wordt van security tools.

Bij het onderzoek naar de security tool wil ik nagaan of een dergelijke tool ook daadwerkelijk een meerwaarde biedt voor bedrijven op het gebied van webontwikkeling. Tools voor dit soort applicaties zijn nog niet zo lang op de markt. De vraag is dan ook of deze tools volwassen genoeg zijn om op grote schaal ingezet te worden tijdens het ontwikkelproces en of de tools goed genoeg zijn om de security van een applicatie er op te kunnen baseren.

5 De securityaspecten van het ontwikkelproces in de praktijk

In dit hoofdstuk neem ik het ontwikkelproces van webapplicaties onder de loep. Hiervoor zal ik de processen van diverse bedrijven vergelijken met de diverse theorieën die zijn opgesteld door de experts. Ik heb er hierbij voor gekozen om te kijken naar kleine en middelgrote bedrijven, mede omdat ik er van uit ga dat bij dit soort bedrijven meer aandacht is voor security tijdens dit proces. Dit baseer ik onder meer op de grote hoeveelheid bronnen die gericht zijn op grote ontwikkelbedrijven en op het feit dat er bij grotere bedrijven een groter budget beschikbaar is om bijvoorbeeld een securityexpert aan te nemen of om security trainingen te geven aan de ontwikkelaars.

Om het ontwikkelproces bij verschillende webdevelopmentbedrijven in kaart te brengen heb ik interviews afgenomen onder vijf bedrijven. Van deze bedrijven heb ik één ontwikkelaar geïnterviewd en gevraagd hoe er binnen hun bedrijf wordt omgegaan met security tijdens het ontwikkelproces. In dit hoofdstuk zoom ik in op de belangrijkste aspecten van deze interviews. De vragenlijst die gebruikt is voor deze interviews vind u in hoofdstuk 10.1 ’Interview’. Het interview is afgenomen bij het bedrijf op locatie omdat dat qua logistiek de makkelijkste oplossing was. Als structuur voor het interview is gekozen voor een open interview waarbij vooral werd doorgevraagd aan de hand van de antwoorden van de ontwikkelaar.

In de sectie 5.1 leest u een samenvatting van de interviews. Deze samenvatting is gemaakt naar aanleiding van de antwoorden van de geïnterviewde. In de volgende secties zoom ik verder in op bepaalde onderwerpen uit deze samenvatting. Deze onderwerpen zal ik naar aanleiding van de literatuur die ik bestudeerd heb vervolgens analyseren. In hoofdstuk 7 ‘Best practices’ zullen de best practices van de interviews en de literatuur worden gecombineerd en verder worden uitgewerkt.

5.1 Samenvatting interviews

Om een goed beeld te geven van de interviews zal in deze sectie een samenvatting worden gegeven van de afgenomen interviews. Het doel van het afnemen van de interviews is om meer informatie te krijgen over de huidige stand van zaken bij webontwikkelbedrijven op het gebied van de omgang met security gerelateerde problemen bij de ontwikkeling van applicaties.

In sectie 5.2 zal de samenvatting uit deze sectie verder worden uitgewerkt en zal er een link worden gelegd tussen de bevindingen uit de praktijk en de theorie die naar voren is gekomen vanuit de literatuurstudie.

5.1.1 Vooraf

Aan dit onderzoek hebben vijf bedrijven deelgenomen die voornamelijk bezig zijn in de webdevelopmentbranche of intern een ontwikkelafdeling hebben gericht op webdevelopment. Het opleidingsniveau van de deelnemers varieert van MBO tot universitair. Alle deelnemers hebben een opleiding met affiniteit in de informatica gevolgd.

De functieomschrijving van de geïnterviewden bestonden in alle gevallen uit programmeur al dan niet gecombineerd met systeembeheer. In sommige gevallen had de geïnterviewde daarnaast ook andere, niet IT gerelateerde, functies.

De omvang van de bedrijven varieert tussen de vier en honderd veertig personen. Hier is bewust voor gekozen om het verschil tussen kleine en middelgrote bedrijven inzichtelijk te maken. In één geval bestond de ontwikkelafdeling slechts uit één persoon. Op het gebied van techniek gebruikte de geïnterviewde bedrijven alleen JAVA of PHP.

De producten die de bedrijven produceerde varieert tussen diverse webapplicaties zoals Content Management Systemen en aanpassingen van bestaande (opensource) softwarepakketten. Daarnaast waren er bedrijven die één standaard product boden en bedrijven die per klant maatwerk produceerden. Twee van de vijf bedrijven ontwikkelde producten voor gebruik binnen het bedrijf, de overige bedrijven produceerde producten voor klanten.

De geïnterviewde personen hadden allemaal ervaring met computerbeveiliging, variërend van interesse vanuit een hobby tot ervaring door het ontwikkelen van oplossingen voor beveiligproblemen op het werk. Twee van de vijf bedrijven organiseren zelf bijeenkomsten waar ontwikkelaars worden bijgeschoold op het gebied van de laatste trends en ontwikkelingen op het gebied van webdevelopment. Bij deze bijeenkomsten komt ook computerbeveiliging regelmatig aan bod.

Buiten deze bijeenkomsten om was er over het algemeen een regeling waarmee werknemers zelf bijscholing konden krijgen in de vorm van een cursus of training. Dit was in alle gevallen wel de verantwoordelijkheid van de ontwikkelaar zelf en deze ruimte is vrij te besteden, mits vak - gerelateerd.

5.1.2 Tools en processen

Alle vijf de bedrijven maken gebruik van, of denken er over om in de toekomst gebruik te gaan maken van 'security tools' om de veiligheid van hun producten te verbeteren. Van de vijf bedrijven maakten 3 bedrijven gebruik van specifieke tools en was één bedrijf bezig met de invoering hiervan. Eén bedrijf was bezig met de oriëntatie op het gebruik van een tool. Bij de JAVA bedrijven zien we 'PMD' en 'Findbugs' als belangrijke tools, bij de PHP bedrijven zien we het gebruik van 'code coverage'. Daarnaast gebruikten de bedrijven een 'issue tracking systeem' of 'ticketomgeving' om bugs te documenteren. In vrijwel alle gevallen werd dit systeem ook gebruikt om beveiligingsgerelateerde bugs te registreren en documenteren.

Bij alle bedrijven was er een vast proces dat werd aangehouden bij de ontwikkeling van webapplicaties. Er was echter geen bedrijf dat als vast onderdeel van dit proces expliciet computerbeveiliging toepaste. Over het algemeen werd dit wel meegenomen maar verdeeld onder ontwikkeling en testfase. Vooral tijdens de ontwikkeling van het product wordt er rekening gehouden met veiligheid. Tijdens het onderhoud is dit in mindere mate het geval.

Bij twee bedrijven werd gebruik gemaakt van (delen van) applicaties geschreven door derden. Deze applicaties worden vooraf getest op allerlei zaken, waaronder veiligheidsaspecten. Als de derde partij een hernieuwde versie uitbrengt zal deze worden gedistribueerd onder de klanten. Het is de verantwoording van de derde partij om hun software up-to-date te houden.

Van de vijf geïnterviewde bedrijven gaven er drie aan dat voor ontwikkelaars binnen het bedrijf twee soorten rollen te onderscheiden waren, namelijk onderhoud en ontwikkeling. Onderhoud richt zich op het onderhouden van bestaande (web)applicaties Ontwikkeling richt zich op het ontwikkelen van nieuwe (web)applicaties. Eén ander bedrijf gaf aan dat ontwikkelaars binnen hun bedrijf drie verschillende rollen kunnen hebben; ontwikkeling, beheer en 'audit'. Per project vervullen ontwikkelaars verschillende rollen zodat ontwikkelaars alle drie de rollen regelmatig vervullen. Volgens het bedrijf heeft dit als voordeel dat een ontwikkelaar ook ziet wat er in een audit gevonden wordt en welke problemen er tijdens het beheer van een applicatie op kunnen treden. Het vijfde bedrijf gaf aan geen gebruik te maken van verschillende rollen.

Een aantal bedrijven gaf expliciet aan een aparte omgevingen te hanteren voor de ontwikkeling, waarop getest wordt en waarop producten daadwerkelijk draaien voor klanten. Dit alles om de code beter af te schermen van de eindgebruikers tijdens het ontwikkelen en testen van applicaties. Eén ontwikkelaar gaf aan dat er 12 maanden geleden een ontwikkelslag is gemaakt gericht op het verbeteren van de veiligheid binnen hun applicatie.

Op het gebied van het meldingsproces van bugs was er bij geen van de bedrijven een apart proces voor beveiligingsgerelateerde bugs. Een aantal ontwikkelaars gaf aan dat voor de gebruikers alle bugs gewoon bugs zijn en dus geen onderscheid (kunnen) maken tussen beveiliging- en niet-beveiliging gerelateerd bugs. Beveiligingsgerelateerde bugs worden over het algemeen doorgegeven aan de ontwikkelaar die verantwoordelijk is voor de code waarin de bug gevonden is, het is vervolgens aan de ontwikkelaar om deze bug te verhelpen.

5.1.3 Ontwikkelmethode en beheer

De methoden die de vijf bedrijven gebruikten tijdens het ontwikkelen van hun webapplicaties waren verschillend per bedrijf. Dit komt onder meer door de omvang van het bedrijf. De ontwikkelaars van de grotere bedrijven gaven aan dat er bij het bedrijf tijd besteed was aan het vastleggen van deze methoden. Bij de kleinere bedrijven werd er meer nadruk gelegd op de ontwikkelaar zelf. Daarnaast was er een verschil te zien tussen de bedrijven die één vast product leveren en de bedrijven die maatwerk leveren. Bij de bedrijven met één product werden er meer technieken gebruikt die voor structurele veiligheid zorgen, terwijl bij de andere bedrijven meer nadruk wordt gelegd op de veiligheidsaspecten die kenmerkend zijn voor het desbetreffende product, bijvoorbeeld het verwerken van creditcard gegevens.

Twee van de geïnterviewde gaven aan dat er ook vanuit andere instanties regelgeving is opgesteld op het gebied van beveiliging. Hierbij moet onder andere gedacht worden aan de 'Wet Bescherming Persoonsgegevens' en de 'Payment Card Industry Stadandard'. Deze regels worden opgelegd om een bepaald niveau van beveiliging af te dwingen als er wordt omgegaan met de gegevens die van toepassing zijn op deze wetten en standaarden.

Bij de kleinere bedrijven valt het verder op dat het hebben van een testtraject of code-reviewsysteem of ontbreekt of niet wordt gehandhaafd omdat hiervoor de tijd of capaciteit niet toereikend is.

Als we kijken naar de eigenschappen van de producten die de bedrijven ontwikkelen, dan zien we dat verschillende productvormen ook verschillende beveiligingsmaatregelen vereisen. Het bouwen van een modulair systeem waarop derde partijen zelfstandig extra modules kunnen bouwen bevat andere beveiligingsaspecten dan een product die deze functionaliteit niet heeft.

Een ander aspect is de openheid van code. Zo zien we dat een partij die bijvoorbeeld hun volledige product open aanbieden aan de klant andere intenties heeft dan een bedrijf dat al de door hun geproduceerde producten in hun eigen beheerde omgeving draaien. Welke vorm hiervan het beste gebruikt kan worden is niet zozeer een kwestie van per definitie beter of slechter, maar hangt erg af van de situatie en de instelling van het bedrijf en zijn klanten.

Met deze keuze komt echter ook het beheer van de applicatie in opspraak. Bij bedrijven die de code open aanbieden aan de klant is het beheer niet noodzakelijk ook een onderdeel van de levering van het product. Indien de applicaties binnen de eigen omgeving worden gehouden is dit echter wel het geval. Op het gebied van veiligheid biedt dit een aantal voordelen zoals het bijhouden van beveiligingsproblemen (logging), monitoren op aanvallen en verhelpen van mogelijke security problemen. In de gevallen dat er over beheer gesproken wordt is dit meestal in combinatie met een 'Service Level Agreement'(SLA) waarin al dan niet gesproken wordt over beveiliging.

De bronnen die binnen de geïnterviewde bedrijven worden geraadpleegd om beveiligingsgerelateerde problemen op te lossen of om up-to-date te blijven op het gebied van de laatste ontwikkelingen op het gebied van beveiliging verschillen per bedrijf. Sommige bedrijven beschikken over een eigen documentatiesysteem waarmee ontwikkelaars elkaar onderling kunnen informeren over problemen. Andere bedrijven gebruiken uitsluitend externe bronnen zoals RSS-feeds of websites. De diverse thema-avonden die bij sommige bedrijven worden gehouden voor de

ontwikkelaars vormen ook een belangrijke bron van informatie, zij het dat deze niet altijd beveiliging gerelateerd hoeven te zijn.

Opvallend genoeg was er bij geen van de bedrijven een budget specifiek voor computerbeveiliging aanwezig. In veel gevallen was er wel een overkoepelend budget, bijvoorbeeld voor research & development, waaruit ook computerbeveiliging gerelateerde zaken bekostigd kunnen worden.

5.1.4 Het beveiligingsprobleem van webapplicaties

Over het beveiligingsprobleem met betrekking tot webapplicaties waren alle geïnterviewden duidelijk; het probleem zit hem in de authenticatie en verificatie van de gebruiker. Het grootste verschil met de klassieke desktopapplicaties is volgens de ontwikkelaars dat webapplicaties beschikbaar zijn voor meerdere, al dan niet geautoriseerde, gebruikers van een netwerk of het internet. Hierdoor speelt beveiliging een grote rol, aangezien het voor kwaadwillenden makkelijker is om toegang te krijgen tot een webapplicatie dan een applicatie die alleen op een desktop draait. Daarnaast is het ook zo dat bij grote webapplicaties veel gebruikers en data worden verwerkt, waardoor het aanvallen van een dergelijke applicatie bij voorbaat veel lucratiever is dan wanneer er een desktop applicatie wordt aangevallen met weinig gebruikers en een kleine hoeveelheid data.

Naast deze verschillen zien de ontwikkelaars ook relatief nieuwe problemen die bij klassieke desktopapplicaties geen gevaren waren. Voorbeelden van deze aanvallen zijn bijvoorbeeld ‘SQL injectie’ en ‘Cross Site Scripting’. Aanvallen op webapplicaties komen steeds vaker voor en deze aanvallen worden ook steeds omvangrijker. Er wordt dagelijks wel een nieuwe manier gevonden om misbruik te maken van dit soort problemen.

De geïnterviewden gaven ook mogelijke oplossingen voor deze nieuwe problemen. Onder andere het limiteren van de rechten in SQL, afvangen en filteren van user input en het authenticeren van gebruikers werd door de ontwikkelaars aangedragen als mogelijkheden om het beveiligingsniveau van een webapplicatie te verhogen. Op het gebied van beheer geven sommige ontwikkelaars aan dat het verstandiger is om de applicaties onder eigen beheer te hosten om updates snel te kunnen installeren en de systemen te monitoren op mogelijke aanvallen.

5.1.5 Awareness & verantwoordelijkheid

PHP werd in het verleden vaak aangemerkt als onveilige programmeertaal. Dit is volgens de geïnterviewde PHP-ontwikkelaars niet zozeer de schuld van de taal maar van de programmeur. Omdat PHP een laagdrempelige programmeertaal is kunnen ook mensen die minder bekend zijn met het programmeren toch aan de slag om een webapplicatie te produceren. Echter het gevaar hiervan is dat deze personen minder bekend zijn met de beveiligingsproblemen van webapplicaties en dat er vaak code wordt overgenomen van derde zonder dat bekend is wat deze code exact doet, dit met vele, veiligheidsgerelateerde, problemen tot gevolg.

Bekendheid met de materie is, zo geven de geïnterviewden aan, een van de grootste problemen op het gebied van computerbeveiliging. In veel gevallen is het zo dat, als alles goed gaat, er geen noodzaak is om (uitgebreide) veiligheidsmaatregelen te nemen. Pas als er iets mis gaat wordt de noodzaak van de beveiligingsmaatregelen duidelijk en is er wel ruimte en tijd voor het nemen van maatregelen. In één geval was het bedrijf wel duidelijk bezig geweest met beveiliging door een ontwikkelslag te maken die specifiek gericht was op het opsporen en oplossen van veiligheidsproblemen. Deze ontwikkelslag was niet zozeer gericht op het oplossen van een specifiek beveiligingsprobleem maar was bedoeld om het complete niveau van security te testen en verbeteren.

Op het gebied van verantwoordelijkheid van de broncode van applicaties is in alle gevallen de ontwikkelaar voor zijn code verantwoordelijk. In een aantal gevallen is er daarnaast ook een leidinggevende die de verantwoordelijkheid draagt voor het totaalproduct. De redenering dat een ontwikkelaar zelf verantwoordelijk is voor zijn eigen code impliceert wel dat een ontwikkelaar voldoende kennis heeft om hiervoor verantwoordelijk te kunnen zijn. Tot slot, sommige van de geïnterviewde ontwikkelaars geven aan dat er een probleem is betreffende de bekendheid en oplettendheid van de leidinggevenden en/of het management. Er is weinig begrip voor extra tijd en/of kosten die gemaakt moeten worden om beveiligingsgerelateerde problemen te vinden en op te lossen. Bij de leidinggevenden overheerst vooral het idee dat problemen lastig te vinden zijn voor aanvallers en wanen zich veilig.

5.1.6 Verwachtingen en toekomstontwikkeling

Als we kijken naar de verwachtingen die de geïnterviewden hebben met betrekking tot computerbeveiliging in de webontwikkeling dan zien we dat er een redelijk uniform antwoord naar voren komt: Webdevelopment wordt steeds volwassener, waardoor ook computerbeveiliging een steeds beter behandeld onderwerp wordt en steeds meer aandacht krijgt. De algemene mening is dat het schrijven van veilige applicaties niet zozeer een taak is van de taal waarin het geschreven is maar van de ontwikkelaar. Ondanks dat sommige talen als veiliger bekend staan dan andere, hangt de veiligheid van een applicatie af van de zwakste schakel, wat in de toekomst steeds meer zal neerkomen op de deskundigheid van de ontwikkelaar.

Daarnaast is de verwachting dat het thema computerbeveiliging altijd een terugkerend thema zal blijven. Vooral omdat er dagelijks nieuwe technieken en systemen worden gebouwd en er regelmatig nieuwe aanvalstechnieken worden gevonden. Een van de ontwikkelingen die genoemd is, is ‘cloud computing’, een systeem waarbij computerbeveiliging een belangrijke rol zal gaan spelen.

Als verbeterpunten voor zowel zichzelf als voor het bedrijf gaven de geïnterviewden vooral aan dat er meer aandacht moet worden besteed aan het trainen van de ontwikkelaars en dat het management er meer bewust van moet zijn dat security een belangrijk onderdeel van het ontwikkelproces moet zijn. Daarnaast gaven de geïnterviewden op de vraag of zij interesse hebben in het inzetten van ‘source code analyse tools’ aan de ontwikkeling van dit soort tools een goede zaak te vinden, maar dat deze alleen ter ondersteuning van het huidige beveiligingsbeleid kunnen worden ingezet, niet ter vervanging.

5.2 Analyse en de praktijk

In deze sectie zal ik bepaalde elementen die uit de interviews zijn voortgekomen verder uitwerken, toelichten en proberen te linken aan de theorie uit de literatuurstudie. Aan bod komen onder meer de verschillende processtrategieën, awareness en diverse tools die te maken hebben met security. In sectie 5.3 zal ik vervolgens de conclusie over de interviews verder uitwerken.

5.2.1 Processtrategiën vergeleken

Uit de interviews is gebleken dat processtrategieën van een aantal zaken afhankelijk zijn. Ten eerste is er de omvang van het bedrijf; in een klein bedrijf is het lastig om security op de kaart te zetten omdat er veel minder middelen beschikbaar zijn zoals tijd en geld. Hierdoor zie je dat zaken als security in het ontwikkelproces niet altijd de aandacht krijgt die het zou moeten verdienen, deze aandacht blijft meestal beperkt tot wat de ontwikkelaar weet en kan, eventuele fouten blijven onopgemerkt door het gebrek aan controle.

Bij de grotere bedrijven zie je dat er meer aandacht besteedt wordt aan security, voorbeelden hiervan zijn bijvoorbeeld ontwikkelslagen speciaal gericht op het verhogen van de security binnen een specifieke applicatie. Daarnaast bieden deze bedrijven meer voorzieningen aan de ontwikkelaars om hun kennis over security up-to-date te houden doormiddel van het organiseren van speciale bijeenkomsten en het aanbieden van cursussen.

Verschillede strategieën bij ontwikkelbedrijven hebben niet zozeer te maken met de grootte van het bedrijf, maar met de producten die het bedrijf ontwikkeld. Een bedrijf dat één specifieke applicatie ontwikkeld en verkoopt, heeft in de praktijk een strategie om dit product zo goed mogelijk te laten functioneren. Hierbij speelt ook security een belangrijke rol. Bij bedrijven die meerdere applicaties leveren en/of maatwerk leveren aan de klant speelt security een kleinere rol, mede door het gebrek aan tijd tijdens het ontwikkelproces.

Voor grotere bedrijven zijn strategieën zoals de Microsoft ‘Secure Development Lifecycle’ of Citigal’s ‘Touchpoint process’ goed gestructureerde en uitgewerkte oplossingen om security onderdeel te laten uitmaken van het ontwikkelproces. Deze strategieën zijn echter niet geschikt voor de kleinere bedrijven omdat deze teveel middelen als geld, tijd en mankracht vereisen om goed te kunnen werken.

5.2.2 Verantwoordelijkheden

Om een goed security beleid te kunnen voeren moeten er verantwoordelijkheden worden vastgelegd, zodat bij problemen iemand verantwoordelijk kan worden gehouden voor dat deel van de applicatie waar het probleem zich heeft voorgedaan. Tijdens het onderzoek kwam naar voren dat bij veel bedrijven deze verantwoordelijk niet formeel is vastgelegd. In veel gevallen is de ontwikkelaar verantwoordelijk voor zijn eigen code, wat inhoudt dat bij problemen in dat deel van de code de ontwikkelaar wordt aangesproken indien er problemen worden gevonden.

Vooral bij de kleinere bedrijven was naast dat de ontwikkelaar zijn eigen code test en analyseert geen methode aanwezig waarbij de code van een ontwikkelaar gecontroleerd werd op fouten. Dit houdt in dat als er een fout in de code van een ontwikkelaar terecht komt deze dus pas gevonden wordt op het moment dat de applicatie al in productie wordt genomen, met alle gevolgen van dien.

Het niet expliciet vastleggen van verantwoordelijkheden op het gebied van de code, dus wie is verantwoordelijk voor welk deel van de code, kan grote problemen geven bij calamiteiten zoals een massieve aanval of bij ontdekking van een nieuwe exploit. Op dat moment zal er zo snel mogelijk een oplossing moeten worden geleverd aan de klant. Dit kan echter worden vertraagd doordat er nu eerst moeten worden uitgezocht wie verantwoordelijk is voor de code waarin het lek is aangetroffen. Daarnaast is er een probleem bij uitval van een ontwikkelaar. In dat geval is het dus niet duidelijk wie verantwoordelijk is voor de code.

Twee van de vijf geïnterviewde ontwikkelaars gaven aan dat er binnen het bedrijf wel een overkoepelende leidinggevende verantwoordelijk was voor het totale project of applicatie. Dit is al een verbetering, aangezien deze leidinggevende vervolgens kan bijhouden wie waarvoor verantwoordelijk is. Ook hier zien we echter een probleem als deze leidinggevende uitvalt. In hoofdstuk 7 `Best practices` zal er verder in worden gegaan op een mogelijke oplossing voor het vastleggen van verantwoordelijkheden binnen de code.

5.2.3 Awareness

Een ander belangrijk punt dat naar voren kwam tijdens de interviews is dat, hoewel security onder ontwikkelaars een bekend feit is, er bij leidinggevenden nog steeds een taboe rust op het erkennen van security als mogelijk probleem. We zien dit vooral terug bij de verdeling van resources binnen een project. De geïnterviewde ontwikkelaars gaven aan dat er vaak te weinig aandacht wordt besteed aan security tijdens de ontwikkeling van applicaties. Er is echter wel een uitzondering op deze regel, als het gaat om applicaties waarbij security vooraf al een bepaalde rol speelt, zoals systemen waarin creditcard informatie een belangrijke rol speelt, is er wel voldoende aandacht voor security. Daarnaast is het zo dat, wanneer de bedreiging groot genoeg is, bijvoorbeeld door een groot security probleem bij de concurrent of bij een andere afdeling van hetzelfde bedrijf, er doorgaans ook (tijdelijk) meer aandacht voor security is.

Zowel McGraw [1] als Howerd & Lipner [3] beschreven de nadelige gevolgen van het onderschatten van security tijdens het ontwikkelproces. Het grootste gevaar van het onderschatten is volgens deze twee partijen dat, wanneer er tijdens het ontwikkelproces niet voldoende aandacht wordt besteed aan security, er een grote kans bestaat dat er na implementatie van het product er zich nog (ernstige) security bugs zich in het product bevinden.

Hoewel de kans altijd aanwezig is dat er nog security bugs in een geïmplementeerde applicatie zitten, is het oplossen van bugs in een geïmplementeerde applicatie altijd vele malen moeilijker dan wanneer deze tijdens de ontwikkelfase al waren verholpen. In het gunstigste geval zal een gebruiker de bug detecteren en melden en kan het probleem met behulp van een kleine update worden verholpen maar dit zal niet altijd het geval zijn. Wat als de bug wordt gevonden door een kwaadwillende partij? Wat als de gebruiker de bug zelf gebruikt om toegang te krijgen tot bedrijf gevoelige informatie?

Daarnaast is het ook moeilijker om bugs te vinden in (oudere) applicaties die al zijn geïmplementeerd omdat vaak niet exact duidelijk is waar de bug zich bevindt binnen de code. In het gunstigste geval is de verantwoordelijkheid voor de code vastgelegd zoals behandeld in de vorige sectie en kan de verantwoordelijke voor de code de bug detecteren en een patch voor het probleem maken en verspreiden.

5.2.4 Budget, kennismanagement & bug reporting

Uit de afgenomen interviews kwam ook naar voren dat er op het gebied van security geen vaste budgettering was bij de vijf onderzochte bedrijven. De budgettering die over het algemeen voor research & development of voor een ontwikkeltraject van een applicatie werd opgesteld werd wel deels gebruikt voor onderzoek naar security, maar er was geen specifiek budget om security gerelateerd onderzoek te kunnen doen.

Op het gebied van kennismanagement zien we dat de omvang van het bedrijf een rol speelt bij de manier waarop kennis wordt gedeeld tussen de medewerkers. Vooral bij de kleinere bedrijven speelt kennismanagement zich af tussen de ontwikkelaars zelf. Er wordt informeel overleg gepleegd over security gerelateerde zaken en bij vragen wordt de hulp van een collega ingeroepen. Kennis die op deze wijze wordt gedeeld wordt niet gedocumenteerd en het is aan de ontwikkelaar zelf om de kennis bij te houden en te gebruiken.

Bij de grotere bedrijven zien we dat er systematischer wordt omgegaan met kennis. Eén van de geïnterviewde bedrijven maakt bijvoorbeeld gebruik van een Wiki omgeving om kennis tussen de ontwikkelaars te delen en te onderhouden. Een bijkomend voordeel hiervan voor dit bedrijf is dat zij ook hun klanten toegang geven tot (een deel van) deze Wiki zodat klanten ook kunnen profiteren van de kennis van hun leverancier.

Hoewel het hebben van een securitybudget niet direct zal leiden tot een verbetering van de security in geproduceerde applicaties zitten er wel duidelijke voordelen aan het hebben van een dergelijk budget. Niet alleen kan een dergelijk budget ervoor zorgen dat er tijdens het ontwikkelproces voldoende rekening gehouden wordt met security, doordat er vooraf is nagedacht over hoe groot het budget moet zijn. Ook helpt het met het inzichtelijk maken van de ontwikkelkosten voor het management [1].

Om dit te bereiken dient het budget dat te besteden is aan security hiervoor per project opgemaakt te worden, dit omdat per project wellicht andere maatregelen nodig zijn om een bepaald niveau van security te kunnen waarborgen. De inhoud van het budget kan echter wel vooraf opgesteld worden, aangezien de middelen die gebruikt worden om het niveau van security te kunnen handhaven wel bekend zijn. Voorbeelden van gegevens die typisch thuis horen in een dergelijke begroting zijn: Belastbare uren tijdens ontwikkeling, kosten training ontwikkelaars, kosten voor softwarematige analyse tools, kosten voor hardwarematige analyse tools, kosten voor security specifiek testen en kosten voor bugtracking (implementatie).

5.2.5 Source Code Analyzers

Alle vijf de bedrijven maken gebruik van, of denken er over om in de toekomst gebruik te gaan maken van 'security tools' om de veiligheid van hun producten te verbeteren. Van de vijf bedrijven maakten drie bedrijven gebruik van specifieke tools en was één bedrijf bezig met de invoering hiervan. Eén bedrijf was bezig met de oriëntatie op het gebruik van een tool. Bij de JAVA bedrijven zien we 'PMD' en 'Findbugs' als belangrijke tools, bij de PHP bedrijven zien we het gebruik van 'code coverage'. Daarnaast gebruikten de bedrijven een 'issue tracking systeem' of 'ticketomgeving' om bugs te documenteren. In vrijwel alle gevallen werd dit systeem ook gebruikt om beveiligingsgerelateerde bugs te registreren en documenteren.

Source Code Analyzers, of wel SCAs, zijn bedoeld als een aanvulling op een bestaand ontwikkelbeleid In 'The Security Development Lifecycle' [3] wordt onder andere in het hoofdstuk 'Stage 7: Secure Testing Policies' het gebruik van dit soort tools besproken. SCA's worden ingezet om de code die ontwikkeld is te scannen op bekende 'exploits'. Hierdoor zijn SCA's niet zozeer bedoeld om op zichzelf te opereren maar moeten gezien worden als een snel middel om de code te doorzoeken op exploits die al bekend zijn. In hoofdstuk 6 `Tools` wordt een voorbeeld gegeven van het juist inzetten van een SCA tijdens het ontwikkelproces.

PMD[4] is een JAVA SCA en richt zich voornamelijk op het zoeken naar potentiële problemen als:

- Mogelijke bugs, code waarvan wordt vermoed dat het niet veilig is om te gebruiken
- 'Dode' code, code die niet gebruikt wordt
- Suboptimale code, code die niet optimaal geschreven is en verbeterd kan worden
- Over gecompliceerde expressies, expressie die te gecompliceerd zijn en verbeterd kunnen worden
- Herhaling van code, code die vaker gebruikt wordt en dus beter in een functie kan worden opgenomen

PMD kan op twee verschillende manieren gebruikt worden, als losse applicatie in de vorm van een Command Line Interface (CLI) of geïntegreerd in de meeste bekende ontwikkelomgevingen voor JAVA. Doordat de tool zelf ook op JAVA gebaseerd is, is dit platform onafhankelijk te gebruiken.

Findbugs[5] is net als PMD een op Java gebaseerde statische Source Code analyse tool. Het programma is in staat om op JAVA gebaseerde bytecode (gecompileerde class bestanden) te scannen op de aanwezigheid van bugs. Dit heeft al voordeel dat ook gecompileerde bestanden, waarvan de broncode ontbreekt, kunnen worden gescand.

Code coverage is niet zozeer een tool maar een werkwijze. Omdat bij deze werkwijze meestal wel een tool wordt gebruikt heb ik toch besloten om het in dit hoofdstuk op te nemen. Met behulp van Code Coverage wordt tijdens het testtraject van een applicatie beschreven op welk niveau de code wordt getest. Dit is nodig omdat sommige delen van de code wellicht een groter risico lopen om

[4] http://pmd.sourceforge.net/
[5] http://findbugs.sourceforge.net/

lekken te bevatten of aangevallen te worden dan andere delen. Om het testen zo efficiënt mogelijk te maken is het daarom zaak vast te stellen welke methode gebruikt zal gaan worden om een bepaald niveau van testen toe te passen. Door een juiste afweging te maken tussen de verschillende codeonderdelen kan een applicatie dus beter getest worden op de cruciale code, zonder dat dit ten koste gaat van het testen van de andere code.

Vooral binnen de JAVA bedrijven zien we dat er verschillende SCA's worden gebruikt. Hoewel ik mij in dit onderzoek vooral op het ontwikkelen in PHP richt is het toch interessant om te kijken naar de tools die ik tijdens de verschillende interviews ben tegen gekomen. In hoofdstuk 6 'Tools' vind u een uitgebreide beschrijving en onderzoek naar een SCA.

5.3 Conclusie

In dit hoofdstuk is de praktijk van security in het ontwikkelproces van webapplicaties bij vijf bedrijven geanalyseerd en vergeleken met de theorie. Uit het onderzoek blijkt dat de wijze waarop de verschillende bedrijven security verwerken in hun ontwikkelproces met name beïnvloed wordt door de bedrijfsgrootte en leeftijd van het bedrijf. Ook zien we dat de ambities om iets aan het securityprobleem te doen vaak komt vanuit een (kleine) groep ontwikkelaars die dit proberen over te brengen op het management.

Bij de kleinere bedrijven wordt de ontwikkelaar geacht zelf verantwoording te dragen voor het bijhouden van zijn kennis op het gebied van security. Bij deze bedrijven is gebleken dat een budget voor scholing vaak ontbreekt en de ontwikkelaar doormiddel van zelfinteresse en communicatie met collega's zijn kennis up-to-date moet houden. Bij de grotere bedrijven worden cursussen en/of trainingsavonden aangeboden vanuit het bedrijf en neemt het bedrijf meer verantwoordelijkheid in het up-to-date houden van de kennis van de ontwikkelaar.

Op het gebied van code analyse en testen worden bij de grotere bedrijven op dit moment veelal applicaties gebruikt als 'Findbugs' en 'PMD'. Deze tools worden simultaan ingezet naast een analyse door een 2^{e} of soms zelf 3^{e} ontwikkelaar.

In het volgende hoofdstuk zullen een aantal tools beschreven worden die de ontwikkelaar tijdens dit proces kunnen gebruiken, waarna in hoofdstuk 7 de best practices worden gegeven waarin de beste eigenschappen van deze beide hoofdstukken zullen worden beschreven.

6 Tools

Naast het interviewen van ontwikkelaars en het bestuderen van de literatuur heb ik voor het schrijven van dit document ook een Source code analyse tool (SCA) onderzocht die de ontwikkelaar kan ondersteunen tijdens het ontwikkelproces van een applicatie, namelijk Fortify. Om deze applicatie te testen heb ik de sourcecode van twee PHP-applicaties geanalyseerd. Vervolgens heb ik de resultaten van de twee scans geanalyseerd.

In sectie 6.1 van dit hoofdstuk zal ik de opties van de software beschrijven. Deze beschrijving is deels afkomstig uit de handleiding en deels gebaseerd op de omschrijvingen en opties van het programma zelf. In sectie 6.2 wordt het door mij uitgevoerde onderzoek van de sourcecode analyse besproken. In sectie 6.3 worden de resultaten uit drie andere onderzoeken besproken namelijk [10], [11] en [12].

6.1 Fortify 360

Fortify 360 is een SCA ontwikkeld door Fortify Software Inc[6]. Deze tool kan ingezet worden tijdens alle levensfases van een applicatie, de ontwikkel fase, de test fase en nadat de applicatie is geïmplementeerd.

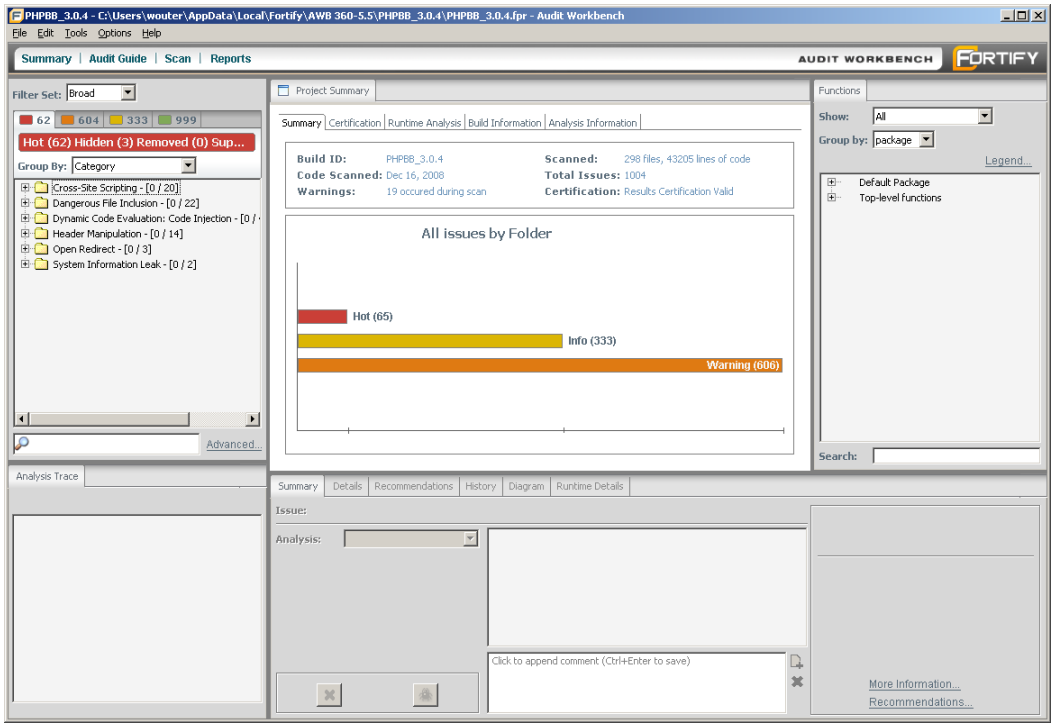
Fortify is verkrijgbaar in drie verschillende vormen, Fortify 360 Analyzers, Fortify 360 en Fortify 360 Server. Deze zijn allemaal verkrijgbaar voor Windows, Linux, Mac en voor Unix zijn alleen de analyzers en server verkrijgbaar, allen in zowel 32 als 64 bit. Het product is in staat om sourecode te analyseren van de talen ASP.NET, C#.NET, VB.NET, C/C++, Java, PLSQL/TSQL, PHP en VB.

[6] http://www.fortify.com/

6.1.1 De software

Het analyzers pakket bevat de command line tools `Source Code Analyzer` (SCA), 'Real Time Analyzer' (RTA) en 'Program Trace Analyzer' (PTA). Met behulp van de SCA kan elke regel code gescand worden op honderden verschillende type kwetsbaarheden. De RTA biedt de mogelijkheid voor het scannen van applicaties die al in productie genomen zijn op mogelijke aanvallen. De PTA wordt gebruikt voor het opsporen van kwetsbaarheden in programma's tijdens de uitvoer. Deze tool wordt voornamelijk gebruikt tijdens de testfase van een project en wordt vaak gebruikt voor een kwaliteit analyse van software.

Het Fortify 360 totaal pakket bevat de SCA, RTA en PTA maar is daarnaast uitgebreid met een 'Audit Workbench' en 'Secure Coding Plugins'. De workbench biedt integratie van de drie command line scanner in één programma met user interface, zodat het voor de ontwikkelaar makkelijker wordt om de code te scannen en de resultaten te analyseren. Deze workbench biedt daarnaast ook de mogelijkheid om middels een module scans te combineren en te vergelijken zodat het scannen van een product verdeeld kan worden over meerdere ontwikkelaars.



1 Fortify 360 Audit Workbench

Met behulp van de 'Secure Coding Plugins' kan de ontwikkelomgeving waarin de programmeur werkt worden uitgebreid met integratie met bijvoorbeeld integratie met een `Bug Tracking System`, de Fortify Workbench en andere security tools. De plugins zijn verkrijgbaar voor de ontwikkelplatformen Eclipse, WebSphere Studio Application Developer en Visual studio 2003 t/m 2008.

De serverversie van Fortify is voornamelijk bedoeld om het security proces te organiseren en monitoren. Met behulp van deze software kan de huidige status van een bepaalde applicatie worden gescanned en kunnen er hulpmiddelen geboden worden aan de ontwikkelaars. Met behulp van Fortify server kan het management daarnaast zien, met behulp van de gestructureerde en duidelijke rapportage functionaliteiten, wat de security status van een applicatie is.

6.1.2 Audit mogelijkheden

Voor het uitvoeren van een audit van sourcecode kan de ontwikkelaar het beste gebruikmaken van de 'Audit Workbench' zodat hij alle scanners gebundeld kan inzetten. Het starten van een audit kan op twee manieren, door te kiezen voor de `Wizard mode` of de `Advanced mode`.

Als de gebruiker kiest voor de 'Wizard mode' krijgt hij vier vragen die beantwoord moeten worden en waarmee het programma de achterliggende scan instellingen bepaald. Deze mode is vooral bedoeld voor gebruikers die zelf niet voldoende security kennis hebben of voor het uitvoegen van een oppervlakkige scan. De vragen die gesteld worden en de bijhorende mogelijke antwoorden zijn:

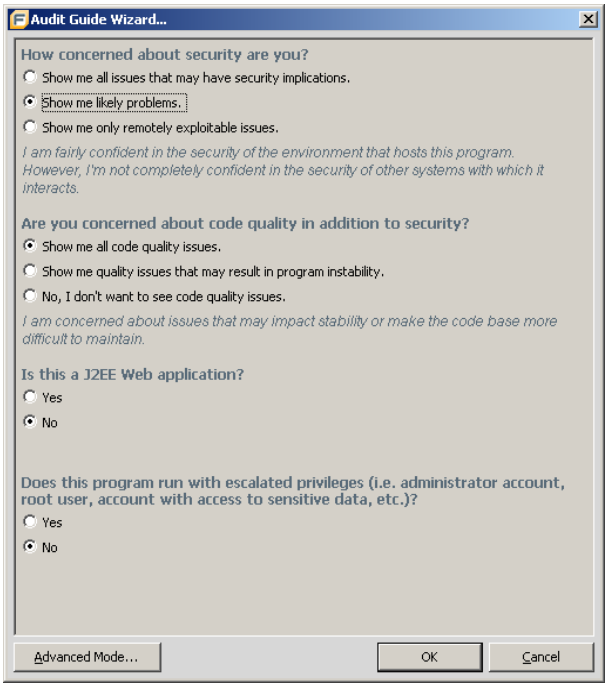
1. **How concerned about security are you?**
   - *Show me all issues that may have security implications (*)*
   - *Show me likely problems*
   - *Show me only remotely exploitable issues*

2. **Are you concerned about code quality in addition to security?**
   - *Show me all code quality issues (*)*
   - *Show me quality issues that may result in program instability*
   - *No, I don't want to see code quality issues*

3. **Is this a J2EE Web application?**
   - *Yes(*)*
   - *No*

4. **Does this program run with escaladed privileges (i.e. administrator account, root user, account with access to sensitive data, etc.)?**
   - *Yes (*)*
   - *No*

* = Deze optie is standaard geselecteerd

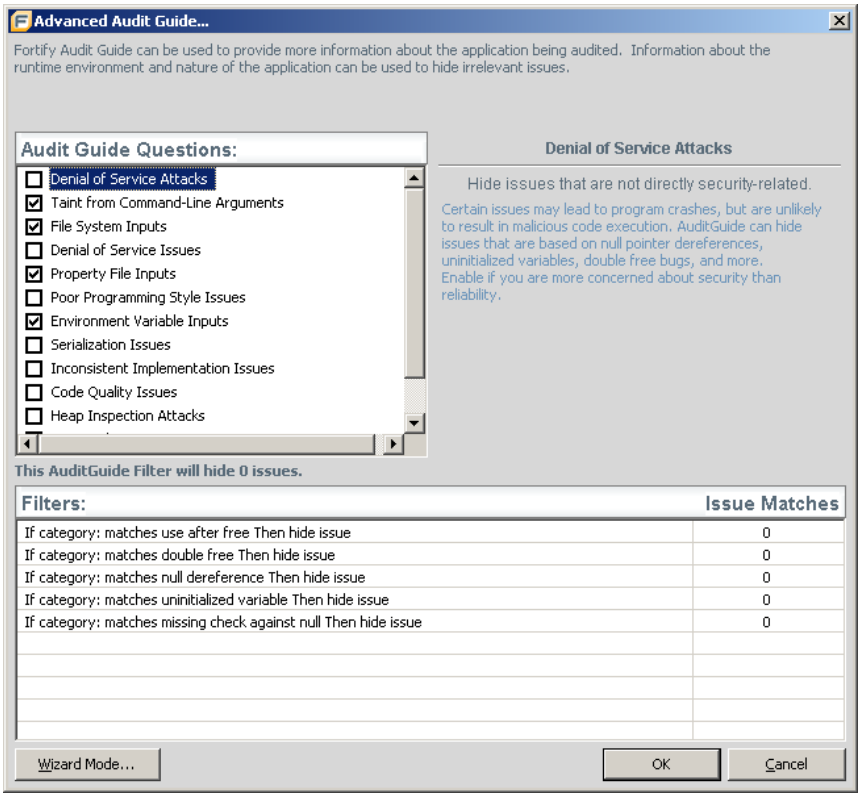


2 Audith Guide Wizard

In de 'Advanced mode' wordt de gebruiker geen vragen gesteld maar krijgt de gebruiker een lijst met mogelijkheden te zien waarop het programma kan scannen. Hoewel er bij elke optie een uitgebreide beschrijving wordt gegeven (zie de lijst hieronder) is voor het gebruik van deze mode wel de nodige security kennis nodig. De lijst met mogelijkheden bestaat uit:

2. *File System Inputs*
3. Denial of Service Attacks
4. Heap Inspection Attacks
5. Environment Variable Inputs
6. Property File Inputs
7. Taint from Command-Line Arguments
8. J2EE Bad Practices
9. Inconsistent Implementation Issues
10. Poor Programming Style Issues
11. Code Quality Issues
12. Database Inputs
13. Serialization Issues
14. Denial of Service Issues

Opvallend is dat de nummering van de selecties niet zichtbaar is in het programma zelf maar wel in de configuratiebestanden. In deze bestanden is de 1e optie niet gedefinieerd maar begint de telling bij 2. Deze opties worden in de volgende pagina weergegeven bij de opties in de 'Wizard' mode, waarbij de optie 1 ontbreekt door bovenstaande reden.



3 Advanced Audit Guide

Nader onderzoek geeft aan dat er een verband is tussen de twee modi, bij het selecteren van een bepaald antwoord in de 'Wizard' mode worden de volgende onderliggende 'Advanced' opties aan of uitgezet:

1. **How concerned about security are you?**
   - *Show me all issues that may have security implications (opties uit: 7, 6, 5, 2, 12)*
   - *Show me likely problems (opties aan: 7, 6, 5, 2 optie uit: 12)*
   - *Show me only remotely exploitable issues (opties aan: 7, 6, 5, 2, 12)*

2. **Are you concerned about code quality in addition to security?**
   - *Show me all code quality issues (opties uit: 3, 10, 11, 13)*
   - *Show me quality issues that may result in program instability (optie aan: 10 opties uit: 3,11,13)*
   - *No, I don't want to see code quality issues (opties aan: 3, 10, 11, 13)*

3. **Is this a J2EE Web application?**
   - *Yes (opties uit: 8)*
   - *No (optie aan: 8)*

4. **Does this program run with escaladed privileges (i.e. administrator account, root user, account with access to sensitive data, etc.)?**
   - *Yes (opties uit: 7, 6, 5, 2)*
   - *No (opties aan: 7, 6, 5, 2)*

De opties 4, 9 en 14 zijn altijd ingeschakeld.

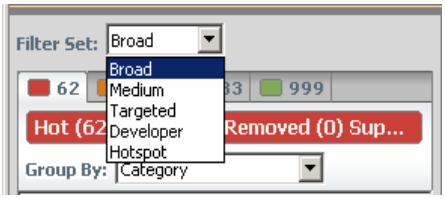
6.1.3 Rulepacks

Naast de opties die je via het programma kunt opgeven bij de start van een audit is het ook mogelijk om 'rulepacks' samen te stellen. Deze packs in de vorm van XML bestanden zijn sets van regels die gebruikt worden tijdens de audit. Fortify levert zelf regelmatig een update van het rulepack die standaard gebruikt wordt bij een audit. Met behulp van de meegeleverde 'Custom Rules Editor' kan de gebruiker zelf regels aanmaken die vervolgens tijdens de audit zullen worden doorlopen. Dit kan handig zijn als je de code wilt scannen op bepaalde kwetsbaarheden die specifiek zijn voor een project zoals Creditcardgegevens.

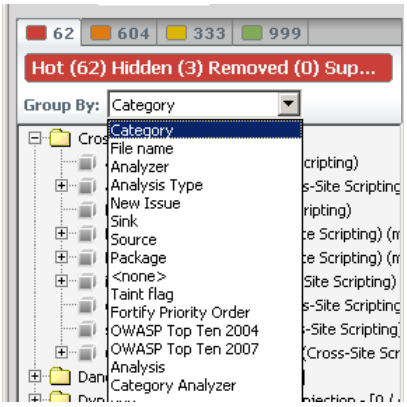
6.1.4 Scanresultaten

Nadat de scanopties zijn ingesteld en de gewenste rulepacks zijn geladen kan het scannen van de code worden uitgevoerd. Nadat de SCA de scan heeft afgerond geeft het programma een overzicht van de gevonden problemen. Deze resultaten worden onderverdeeld in drie hoofdcategorieën; Hot, Warning en Info. Binnen deze hoofdcategorieën zijn vervolgens weer een aantal subcategorieën waarin de problemen worden gegroepeerd zoals Cross-site scripting, Dangerous File Inclusion, Open redirect et cetera.

Met behulp van een 'Filter set' kan het niveau van weergaven op het gebied van de hoofdcategorieën worden aangepast. Deze filter sets filteren op dit niveau de gevonden problemen zodat alleen de relevante problemen worden weergegeven. De problemen die niet relevant zijn voor de gekozen set worden automatisch verborgen. De aanwezige filters zijn; Broad, Medium, Targeted, Developer en Hotspot.



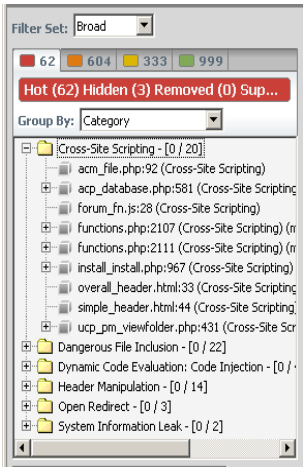
4 Filter Set selectie



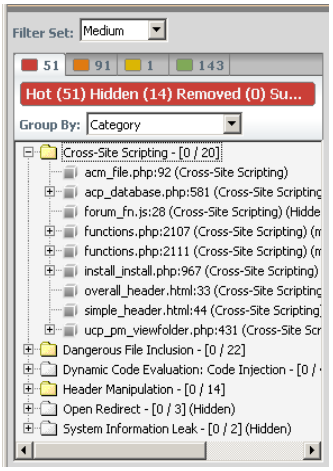
5 Group By selectie

Naast deze filters kan ook op het gebied van de subcategorieën de weergave worden aangepast met behulp van een 'Group by' optie. Deze optie zorgt er niet voor dat er problemen worden verborgen maar zorgt voor een andere groepering van subcategorieën. Een goed voorbeeld van een groepering is bijvoorbeeld de 'OWASP Top Ten 2007' groepering, waarbij de gevonden problemen worden gegroepeerd op basis de lijst met tien meest voorkomende gevaren samengesteld door OWASP. Gevonden problemen die niet kunnen worden gegroepeerd met behulp van de 'group by' optie worden onderverdeeld in een aparte categorie genaamd 'Not Covered'.

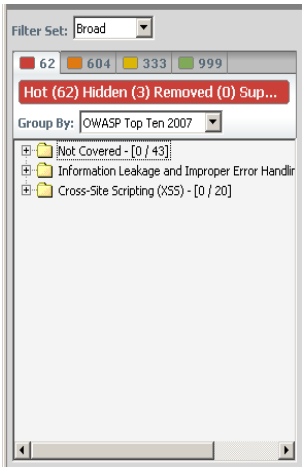
De instelling van het filter geeft directe resultaat op het analyzerapport (zie ook de volgende sectie). Terwijl de 'group by' functionaliteit alleen betrekking heeft op de weergave binnen de applicatie zelf.



7 Category



8 Filter Medium



6 Group by OWASP top ten 2007

6.1.5 Fortify analyse

Zoals we in de vorige sectie zagen kunnen de analyzeresultaten van de code op verschillende manieren worden gepresenteerd. In het programma zelf kan er gekozen worden uit diverse filters waarmee de threats gefilterd worden weergegeven en de gegevens kunnen ook worden geëxporteerd met verschillende templates. Deze filters en templates zijn echter zo opgesteld dat er vooral voor ontwikkelaars een hoop informatie kan worden gehaald uit een analyse van de code. Maar deze informatie kan ook bijvoorbeeld aan het management worden gepresenteerd. Hiervoor kan bijvoorbeeld gebruik gemaakt worden van de mogelijkheid om de codeanalyse weer te geven gegroepeerd op de OWASP top 10 van 2004 of 2007 met behulp van de 'Group by' functionaliteit uit de vorige sectie.

Maar naast deze opties om de data binnen de workbench weer te geven heeft Fortify ook een rapportage functie waarmee rapporten gegenereerd kunnen worden in PDF, XML en RTF formaat met vijf verschillende indelingen. Het 'Foritfy Security Raport' is het algemene rapport van Fortify en bestaat uit zes hoofdstukken: 'Executive Summary', 'Project Summary', 'Results Outline', 'Detailed Project Summary', 'Issue Count By category' en 'Issue Breakdown Analysis'. Daarnaast is er eventueel nog een zevende hoofdstuk genaamd 'New Issues', dit hoofdstuk beschrijft mogelijke nieuwe bugs die gevonden zijn in de code als de code al eerder is geanalyseerd.

Het 'Fortify Developer Workbook' is een rapport speciaal bedoeld voor de ontwikkelaar van de geanalyseerde code en bevat drie secties, te weten: 'Report Overview', 'Issue Summary' en 'Results Outline'. De gegevens betreffende de verschillende threats zijn hierbij weggelaten omdat er van wordt uitgegaan dat de ontwikkelaar deze al kent.

De volgende twee templates zijn de OWASP Top Ten 2004 en OWASP Top Ten 2007 welke beide gelijk zijn qua indeling; 'Report Overview', 'Issue Breakdown by OWASP' en 'Results Outline'. Het enige verschil tussen deze twee rapporten is dat het ene rapport is opgesteld aan de hand van de OWASP top tien van 2004 en de andere op de lijst uit 2007. Het laatste rapport is de 'Fortify Scan Summary' en bestaat uit de hoofdstukken 'Issue Count By Category', 'Project Summary' en 'Detailed Project Summary'. Zoals de hoofdstukken al laten blijken is dit rapport meer geschikt voor het management aangezien het alleen maar samenvattingen bevat van de resultaten.

De Fortify Workbench is dus in staat om op een zeer gedetailleerd niveau de ontwikkelaar feedback te geven na het uitvoeren van een scan, maar het programma kan ook deze data gebruiken om een rapport te genereren voor het management. Dit is nuttig omdat het management niet altijd dezelfde data wil zien als de ontwikkelaar omdat ze niet dezelfde kennis hebben van de gegevens die na een scan worden gegenereerd.

6.2 Fortify in de praktijk

Voor het onderzoek heb ik gebruik gemaakt van de Fortify Workbench, dit omdat deze tool het belangrijkste hulpmiddel van de Fortify programma's is die tijdens de ontwikkeling van een applicatie kan worden ingezet. Voor het analyseren maak ik gebruik van de standaard rulepacks van Fortify.

Omdat het in mijn onderzoek gaat om het scannen van PHP- code heb ik alleen de Source Code Analyzer (SCA) uit het complete pakket getest. Dit is de enige scanner die van toepassing is tijdens een PHP-ontwikkeltraject aangezien de Program Trace Analyzer (PTA) bedoeld is voor tijdens de testfase en de Real-Time Analyzer (RTA) tijdens productiefase.

De projecten die ik heb gekozen om te analyseren zijn een CMS-systeem en een CRM applicatie die ik zelf heb ontwikkeld in de afgelopen drie jaar. Beide systemen zijn gebaseerd op de Smarty Template Engine en geschreven voor gebruik op PHP 4 en 5. Binnen het CMS worden er drie modules gebruikt van derde partijen namelijk Smarty (versie 2.6.20), PCLZip (versie 2.6) en (TinyMCE versie 3.0.5)

Beide projecten zijn tijdens het onderzoek met de volgende instellingen gescand:

1. Standaard scan via de 'Wizard mode' met de volgende antwoorden geselecteerd
    - **How concerned about security are you?**
        - *Show me all issues that may have security implications*
    - **Are you concerned about code quality in addition to security?**
        - *Show me all code quality issues*
    - **Is this a J2EE Web application?**
        - *No*
    - **Does this program run with escaladed privileges (i.e. administrator account, root user, account with access to sensitive data, etc.)?**
        - *No*
2. Scan via de advanced mode met alle opties geselecteerd
3. Scan via advanced mode na analyse van de vorige scan waarbij de opties zijn aangepast om zo min mogelijk false positives te krijgen zonder dat er (te)veel positives verloren zullen gaan.

Met deze verschillende scans wil ik kijken of er een duidelijk verschil is tussen de wizard- en advanced mode en of het verstandig is om de geavanceerde mode aan te passen om zo een beter resultaat te krijgen met minder false positives.

6.2.1 CMS resultaten

In totaal bedraagt de volledige omvang van de source code 22713 regels code, verdeeld over 337 bestanden. Fortify had voor de scan in 'Wizard mode' bijna 6 minuten nodig waarbij er 248 threats gevonden zijn. Na analyse van deze threats waarvoor 50 minuten nodig waren bleek dat van de threats er 131 false positives waren (resp. 53%)

Categorie	Issues	% gevonden issues	True positive	False positive	% true positives	% toaal true positives
Hot	**56**	**23%**	**39**	**17**	**70%**	**33%**
Cross Site Scripting	53	21%	36	17	68%	31%
Dangerous File Inclusion	2	1%	2	0	100%	2%
Open Redirect	1	0%	1	0	100%	1%
Warning	**119**	**48%**	**23**	**96**	**19%**	**20%**
Cookie Security: Cookie not Sent Over SSL	5	2%	5	0	100%	4%
Dangerous Function	6	2%	0	6	0%	0%
Password Management: Hardcoded Password	2	1%	0	2	0%	0%
Path Manipulation	65	26%	4	61	6%	3%
Resource Injection	3	1%	0	3	0%	0%
System Information Leak	15	6%	9	6	60%	8%
Weak Cryptographic Hash	23	9%	5	18	22%	4%
Info	**73**	**29%**	**55**	**18**	**75%**	**47%**
Cookie Security: HTTPOnly not Set	5	2%	5	0	100%	4%
Cross-Site Request Forgery	37	15%	37	0	100%	32%
JavaScript Hijacking: Ad Hoc Ajax	8	3%	7	1	88%	6%
JavaScript Hijacking: Vulnerable Framework	6	2%	6	0	100%	5%
Often Misused: File Upload	3	1%	0	3	0%	0%
Password Management: Password in Commen	14	6%	0	14	0%	0%
Totaal	**248**	**100%**	**117**	**131**	**47%**	**100%**

Voor de scan in ‘Advanced mode’ had het programma ook bijna 6 minuten nodig waarbij er ook 248 threats gevonden werden waarvan 131 false positives (resp. 47%), de scanresultaten waren exact hetzelfde als bij de wizard mode, waarmee ik direct kan concluderen dat de scanoptie ‘J2EE Bad Practices’ geen invloed heeft op het resultaat van de scan aangezien dit het enige verschil is tussen beide scans. Dit is overigens niet vreemd omdat deze optie alleen van toepassing is op JAVA code.

Voor de laatste scan heb ik een aantal varianten van de opties in de ‘Advanced mode’ geprobeerd, maar het aantal threats bleef altijd gelijk. Nader onderzoek geeft aan dat deze opties eigenlijk alleen geschikt zijn voor het scannen van JAVA applicaties, het aanpassen van de scanregels voor PHP-applicaties ontbreekt helaas nog.

Een uitgebreide analyse van de verschillende categorieën en scanresultaten vind u in sectie 6.2.5

6.2.2 CRM Resultaten

De sourcecode van dit project bestaat uit 4082 regels code, verdeeld over 24 bestanden, waarbij dit alleen code is die voor deze applicatie is geschreven en waarbij er geen gebruik is gemaakt van code van derden. Voor de scan had het programma bijna 2 minuten nodig om alle code te doorlopen waarbij het programma 173 threats heeft gedetecteerd waarvan 8 false positives (5%). Ook bij dit project was er geen verschil tussen de 'Wizard mode' en de 'Advanced mode'.

Categorie	Issues	% gevonden issues	True positive	False positive	% true positives	% toaal true positives
Hot	**145**	**84%**	**142**	**3**	**98%**	**86%**
Cross Site Scripting	1	1%	0	1	0%	0%
Dangerous Functions	1	1%	0	1	0%	0%
Header Manipulation: Cookies	1	1%	0	1	0%	0%
SQL Injection	142	82%	142	0	100%	86%
Warning	**15**	**9%**	**13**	**2**	**87%**	**8%**
Cookie Security: Cookie not Sent Over SSL	4	2%	4	0	100%	2%
Path Manipulation	4	2%	2	2	50%	1%
System Information Leak	1	1%	1	0	100%	1%
Weak Cryptographic Hash	6	3%	6	0	100%	4%
Info	**13**	**8%**	**10**	**3**	**77%**	**6%**
Cookie Security: HTTPOnly not Set	4	2%	4	0	100%	2%
Cookie Security: Overly Broad Path	3	2%	3	0	100%	2%
Cookie Security: Persistent Cookie	3	2%	3	0	100%	2%
Password Management: Password in Comment	3	2%	0	3	0%	0%
Totaal	**173**	**100%**	**165**	**8**	**95%**	**100%**

De sourcecode van de CRM-applicatie verschildt in complexiteit van de sourcecode van de CMS-applicatie mede omdat de CMS-applicatie later geschreven is waardoor ik beter bekend was met de code. De laatste aanpassingen aan het CRM systeem zijn gemaakt eind 2006, ruim twee jaar geleden. Dit is goed te merken in het analyseren van de fouten, veel code kwam mij niet direct bekend voor en daardoor moest ik vaak een paar stappen terug doen om te kijken waar de code nu precies voor gebruikt werd en wat het probleem eventueel kon afvangen. Dit resulteert in een handmatige analysetijd van 2 uur en 25 minuten, bijna drie keer zo lang als bij de analyse van de CMS code. Een uitgebreide analyse van de verschillende categorieën en scanresultaten vindt u in sectie 6.2.5.

6.2.3 Resultaten nader bekeken

Als we de resultaten van beide scans nader bestuderen dan zien we dat Fortify in bepaalde categorieën een beduidend hoger percentage false positives heeft gevonden dan in andere categorieën.

Categorie	Issues CMS	Issues CRM	Issues Total	% gevonden issues	True positive	False positive	% true positives	% toaal true positives
Hot	**56**	**145**	**201**	**48%**	**181**	**20**	**90%**	**64%**
Cross Site Scripting	53	1	54	13%	36	18	67%	13%
Dangerous File Inclusion	2	0	2	0%	2	0	100%	1%
Open Redirect	1	0	1	0%	1	0	100%	0%
Dangerous Functions	0	1	1	0%	0	1	0%	0%
Header Manipulation: Cookies	0	1	1	0%	0	1	0%	0%
SQL Injection	0	142	142	34%	142	0	100%	50%
Warning	**119**	**15**	**134**	**32%**	**36**	**98**	**27%**	**13%**
Cookie Security: Cookie not Sent Over SSL	5	4	9	2%	9	0	100%	3%
Dangerous Function	6	0	6	1%	0	6	0%	0%
Password Management: Hardcoded Password	2	0	2	0%	0	2	0%	0%
Path Manipulation	65	4	69	16%	6	63	9%	2%
Resource Injection	3	0	3	1%	0	3	0%	0%
System Information Leak	15	1	16	4%	10	6	63%	4%
Weak Cryptographic Hash	23	6	29	7%	11	18	38%	4%
Info	**73**	**13**	**86**	**20%**	**65**	**21**	**76%**	**23%**
Cookie Security: HTTPOnly not Set	5	4	9	2%	9	0	100%	3%
Cookie Security: Overly Broad Path	0	3	3	1%	3	0	100%	1%
Cookie Security: Persistent Cookie	0	3	3	1%	3	0	100%	1%
Cross-Site Request Forgery	37	0	37	9%	37	0	100%	13%
JavaScript Hijacking: Ad Hoc Ajax	8	0	8	2%	7	1	88%	2%
JavaScript Hijacking: Vulnerable Framework	6	0	6	1%	6	0	100%	2%
Often Misused: File Upload	3	0	3	1%	0	3	0%	0%
Password Management: Password in Comment	14	3	17	4%	0	17	0%	0%
Totaal			**421**	**100%**	**282**	**139**	**67%**	**100%**

Kijken we naar de hoofdcategorieën dan zien we dat 'Hot' het grootste aantal gevonden problemen bevat en tevens ook het hoogste aantal true positives (181). Bij de categorie 'warnings' is het percentage gevonden problemen bijna een derde van het totaal aantal gevonden problemen, maar slechts 27% hiervan is ook daadwerkelijk true positive wat een totaal geeft van slechts 13% van het totaal aantal issues.

Als we naar de subcategorieën kijken zien we dat er sommige subcategorieën zijn waarin Fortify beduidend slechter scoort dan andere categorieën op het gebied van het detecteren van true positives. Vooral de categorieën 'Path manipulation' en 'Weak Crypthographic Hash' waarin toch volgens het programma veel problemen in zijn gedetecteerd hebben een percentage true positives van minder dan 40%.

Na verdere analyse is duidelijk geworden dat bij bijvoorbeeld 'Weak Crypthographic Hash' het programma op zich goed in staat is om crypthographische hashfuncties als Message Digest Algorithm 5 (MD5) te detecteren, maar dat hij vervolgens wel ieder gevonden functie als issue bestempeld. Dit is naar mijn mening niet correct, een dergelijke functie kan ook gebruikt worden voor niet security gerelateerde zaken zoals om een unieke bestandsnaam te genereren. Fortify is blijkbaar goed in het vinden van deze functies, maar minder goed in het beoordelen van het correcte gebruik hiervan.

Bij het analyseren van de problemen kwam ik ook tot de conclusie dat niet iedere categorie even zinvol is om op te nemen in een eventuele verdere analyse. Een voorbeeld hiervan is de categorie 'Password Management: Password in Comment'. Deze categorie betreft problemen die aanduiden dat er in het commentaar bij het script het woordje 'password' is opgenomen. Dit hoeft uiteraard geen enkel security gerelateerd probleem op te leveren, mits er geen wachtwoorden 'hard coded' zijn opgenomen in de code. Echter voor die detectie is een aparte categorie 'Password Management: Hardcoded Password' die dit probleem eventueel al meldt.

Bij andere categorieën zoals 'Cross-Site Scripting' (CSS) en 'Cross-Site Request Forgery' (CSRF) viel het op dat in mijn geval dezelfde waarschuwingen erg vaak terug komen. Dit is aan de ene kant erg onhandig omdat steeds dezelfde oplossing gebruikt kan worden en je dus moet nagaan of het probleem echt hetzelfde was als de vorige x aantal oplossingen. Aan de andere kant is het ook wel gewenst, aangezien je nu wel gewezen wordt op het feit dat hetzelfde probleem zich vaker voordoet op verschillende plaatsen binnen de geanalyseerde sourcecode.

Uiteindelijk blijkt dat van de 421 problemen die aanvankelijk door Fortify gedetecteerd zijn er 282 (67%) daadwerkelijk echt een probleem vormen voor de gescande applicaties. Naar aanleiding van deze analyse zou ik aanraden om in ieder geval te kijken naar de problemen in de categorieën 'Cross Site Scripting', 'SQL Injection' en 'Cross-Site Request Forgery' welke samen 76% van het totaal aantal positives vormen.

6.2.4 Verfijning van de resultaten

Nadat de resultaten zijn geanalyseerd wil ik de weergaven van het gevonden aantal issues verwerken door de weergave van de resultaten binnen Fortify te wijzigen. Het doel hiervan is om bijvoorbeeld de gegevens aan ontwikkelaars door te geven zonder dat hij irrelevante informatie zoals categorieën met een klein aantal true positives zelf moet scheiden van de belangrijkere problemen.

Zoals ik eerder aangaf levert het vooraf wijzigen van de audit instellingen niet het gewenste resultaat op, de gevonden problemen zijn niet afhankelijk door de gemaakt instellingen. Maar Fortify biedt de mogelijkheid om zelf de weergave van hoofd- & subcategorieën aan te passen (zie ook sectie 6.1.4)

Om te kunnen testen in hoeverre Fortify aanpasbaar is aan de wens van de gebruiker heb ik de grens van het aantal true positives ten opzichte van het totaal aantal true positives op 80% gesteld. Zaak is hierbij dat er zoveel mogelijk problemen worden opgelost zonder dat dit ten koste gaat van het aantal positive problemen dat gevonden is. De volgende subcategorieën voldoen aan deze voorwaarden:

Categorie	Issues Total	% gevonden issues	True positive	False positive	% true positives	% toaal true positives
Cross Site Scripting	54	21%	36	18	67%	13%
SQL Injection	142	54%	142	0	100%	50%
Weak Cryptographic Hash	29	11%	11	18	38%	4%
Cross-Site Request Forgery	37	14%	37	0	100%	13%
Totaal	**262**	**100%**	**226**	**36**	**86%**	**80%**

Om deze vier categorieën weer te geven kan ik de interface op twee manieren aanpassen, met behulp van de filters of de groepeerfunctie. Het kiezen van een ander filter levert echter niet het gewenste resultaat op, geen van de filters geeft de informatie uit deze vier categorieën. Ook met behulp van de “Group by” functionaliteit kan de weergave niet worden ingesteld op het juiste niveau. Hoewel er wel een optie is om bijvoorbeeld alleen de hoofdsectie hot weer te geven is er geen mogelijkheid om een schifting te maken tussen de subcategorieën.

6.3 Bevindingen eerder onderzoek

Ware & Fox [10] beschreven in hun paper hun onderzoek naar de correctheid van statische analyse tools waaronder Fortify. In hun onderzoek beschreven zij 115 verschillende problemen binnen de JAVA secure code heuristiek, voor het verhogen van de kwaliteit en veiligheid van JAVA code, welke zij hebben getest met behulp van acht verschillende tools. Omdat er voor JAVA nog geen secure code heuristiek was gedefinieerd hebben Ware en Fox voorafgaand aan hun onderzoek een totale lijst met mogelijke problemen die zich kunnen voordoen binnen de JAVA source code opgemaakt. Hun conclusie uit dit onderzoek is dat, als de tools ingesteld stonden op hun standaard instellingen, het nog steeds nodig was dat er een menselijke code review plaats vond voor het detecteren van alle problemen. Ze vermelden hier wel bij dat ze er vanuit gaan dat deze menselijke code review alle problemen zal vinden, dus ook die problemen die de tools niet konden identificeren.

Het onderzoek van Ayewah & Pugh [11] is niet zozeer gericht op de werking van Sourcecode Analyzers maar meer op de gebruikers van deze tools. Uit hun onderzoek blijkt dat verschillende type gebruikers geïnteresseerd zijn in verschillende niveaus van meldingen. Zelfs in de laagste categorieën, dus problemen met de minste impact, zijn er nog steeds groepen gebruikers te vinden die deze meldingen als een gevaar beschouwen. Hierdoor kan volgens de onderzoekers het gevaar bestaan dat belangrijkere problemen niet worden opgelost omdat de ontwikkelaars bezig zijn met het oplossen van problemen met een veel lagere impact. De onderzoekers schrijven verder dat naar hun mening het opstellen van regels betreffende het gebruik van sourcecode analyse tools binnen bedrijven dit probleem zou kunnen verkleinen dan wel wegnemen.

Strijbos [12] beschrijft in zijn master thesis het classificeren van webapplicaties op het gebied van security. De achterliggende gedachte hierbij is dat gebruikers van webapplicaties aan de hand van het certificaat kunnen zien hoe veilig het programma is. Als mogelijk middel van het classificeren van de applicaties stelt Strijbos het gebruik van sourcecode analyse tools voor, waaronder Fortify. Uit zijn onderzoek bleek dat het mogelijk is om de SCA's te gebruiken om een certificeringproces op te zetten maar dat de SCA's op een aantal aspecten zullen moeten worden aangepast. Als belangrijkste verbeterpunten noemt hij onder meer de mogelijkheden voor het scannen van webapplicaties, het onvermogen van de scanner bij het vinden van lastige problemen en het feit dat de SCA's teveel zinloze kwetsbaarheden detecteren.

6.4 Conclusie

Fortify 360 is een compleet pakket voor de gehele levensduur van een applicatie. Door de compatibiliteit met diverse programmeertalen als ASP.NET, C#.NET, VB.NET, C/C++, JAVA, PLSQL/TSQL, PHP en VB is het programma breed inzetbaar. Voor het scannen van een op PHP gebaseerde applicatie blijkt de scanner echter gelimiteerd te worden door het feit dat er voor deze programmeertaal, en wellicht is PHP hierin niet de enige, toch een beperkt aantal opties mogelijk zijn ten opzichte van het scannen van een JAVA project. Zo is het niet mogelijk om de opties van een scan te verfijnen, waardoor het verschil tussen de 'wizard' en 'geavanceerde' scan compleet wegvalt.

Ondanks deze constatering is de software nog steeds een goede aanvulling op het handmatig analyseren van sourcecode. Niet zozeer omdat het programma beter is in het opsporen van foutgevoelige code maar wel omdat het deze taak snel en relatief goedkoop uitvoert. Jeff Atwood [22] schrijft in zijn artikel over het feit dat de kosten voor hardware steeds verder afnemen terwijl de kosten voor het inhuren van een ontwikkelaar toenemen, ditzelfde geldt voor software als Fortify. Ter vergelijking, waar de aanschaf van Fortify rond de $2.000, - per jaar ligt, kost een ontwikkelaar per jaar gemiddeld $ 87.000, - [7]

Dit wil echter niet zeggen dat Fortify een analyse door een programmeur kan vervangen. Uit voorgaand onderzoek [12], waarin een aantal concurrerende softwarepakketten getest en vergeleken werden, blijkt de software al veel kwetsbaarheden te detecteren maar dat er nog steeds gevallen zijn die niet gedetecteerd worden. Daarnaast zullen fouten die geconstateerd zijn door Fortify door een programmeur moeten worden nagegaan op false positives, ook dit zal in de toekomst nodig blijven.

Dit laatste probleem kan echter wel beter beheersbaar worden gemaakt door een aantal maatregelen te nemen en afspraken te maken voordat Fortify wordt ingezet bij een project. Door een volledige analyse te maken van de code en deze analyse na te gaan op false positive meldingen kan er gekeken worden in welke categorieën het nodig is om de software te scannen. Mijn onderzoek gaf aan dat er op dit gebied echter wel de nodige verbeteringen gemaakt moeten worden doorgevoerd binnen Fortify, mede omdat het probleem, zoals beschreven in sectie 6.2.4, op dit moment nog een moeilijk controleerbaar proces bleek te zijn.

Mede daardoor is het programma naar mijn mening niet geschikt om een compleet project van grote omvang geheel te analyseren omdat dit simpelweg teveel false positives zal geven waardoor de effectiviteit van het gebruik van de software verloren gaat in het analyseren van de meldingen die het programma geeft. De software kan effectiever worden ingezet door iedere ontwikkelaar zijn eigen code te laten analyseren, bijvoorbeeld eenmalig per week. Hierdoor blijft de omvang van de code, en dus ook die van de false positives, beperkt en kan de ontwikkelaar ook goed de false positives onderscheiden van de true positives.

[7] http://www.payscale.com/research/US/Job=Sr._Software_Engineer_/_Developer_/_Programmer/Salary

7 Best practices

In dit hoofdstuk zullen de best practices worden besproken die ik middels het literatuuronderzoek, de interviews en het werken met Fortify ben tegengekomen. Ik heb niet de intentie om een complete lijst met alle best practices te geven, aangezien er al diverse literatuur beschikbaar is die dit beschrijft. Het doel is om juist die best practices aan te duiden die in de praktijk bij middelgrote en kleinere ontwikkelbedrijven van belang kunnen zijn.

De best practices die ik zal behandelen zijn opgedeeld in de volgende groepen: 'management & organisatie', 'inrichting van het ontwikkelproces' en tot slot 'wat te doen na implementatie van een product.

7.1 Management & organisatie

Binnen de meeste bedrijven is er sprake van een hiërarchische structuur. Dit is een belangrijk wanneer het neerkomt op verantwoordelijkheid en sturing maar vooral tijdens het ontwikkelproces is het zaak om de lijnen tussen de verschillende lagen binnen de hiërarchie zo kort mogelijk te houden. Dit zorgt er niet alleen voor dat het management of leidinggevende zeer betrokken is bij het project maar ook dat er awareness gecreëerd kan worden op het gebied van security.

Om dit te bereiken is het zaak dat het management of leidinggevende van het project de securityrisico's van een project goed kan inschatten en up-to-date informatie ontvangt over de mogelijke problemen die zich voordoen tijdens de ontwikkeling maar ook na implementatie van een product. Een goed begin van het inschatten van de securityrisico's is het opstellen van een security budget, voor zowel de kosten als tijd die gespendeerd mogen worden aan security gerelateerde onderwerpen als training en onderzoek. Hierbij dient voor dat het project begint een inschatting gemaakt te worden van de securityrisico's van een project, iets wat het management kan helpen om inzage te krijgen in deze risico's.

Uiteraard is awareness niet alleen een zaak die speelt bij het management, ook de ontwikkelaars zelf moeten het besef hebben dat security een belangrijke rol kan spelen tijdens een project. Een manier om dit te bereiken is door te zorgen voor goede opleidingen en cursussen op het gebied van security door deze op te nemen in het securitybudget of door een persoonsgeboden budget per ontwikkelaar vast te stellen. Hierbij kan het gaan om een geldbedrag dat ontwikkelaars kunnen besteden aan security gerelateerde trainingen, maar beter nog is het om een aantal uren te beschrijven die de ontwikkelaar kan spenderen aan het volgen van trainingen. Hierdoor kan de ontwikkelaar zijn kennis op het gebied van security up-to-date houden, iets wat niet alleen zijn programmeervaardigheden ten goede komt maar ook zijn awareness op het gebied van security.

Een andere mogelijkheid die ik ben tegen gekomen tijdens het interviewen van de ontwikkelaars is het rouleren van functie en taken bij diverse projecten. Het proces wat een applicatie doorloopt tijdens zijn levensfase is ruwweg op te delen in drie fasen: de ontwikkelfase, waarin het product wordt ontworpen en gebouwd. De testfase, waarin het product wordt getest op onder andere security aspecten en tot slot de implementatie fase, waarin het product wordt geïmplementeerd en in gebruikt wordt genomen.

Door bij verschillende projecten de taken van de ontwikkelaar te rouleren over deze drie fases wordt de ontwikkelaar betrokken bij het complete proces dat nodig is om een applicatie te ontwikkelen en zal de ontwikkelaar bij het ontwikkelen van een volgende applicatie dan ook gebruik kunnen maken van de opgedane kennis. Daarnaast kan het de productiviteit van een ontwikkelaar ten goede komen omdat bijvoorbeeld het implementatie traject en ontwikkeltraject van twee verschillende projecten naast elkaar kunnen worden doorlopen. Hierdoor krijgt de ontwikkelaar de mogelijkheid om afwisseling in zijn dagelijkse werkzaamheden te creëren.

Er zit echter wel een aantal beperkingen aan het gebruik van deze methode en dat is onder andere dat het ontwikkelteam een minimale grootte moet hebben van 25 ontwikkelaars. Als er minder ontwikkelaars zijn zal het rouleren van de ontwikkeltaken lastig te implementeren zijn, omdat er niet bij elke fase evenveel ontwikkelaars nodig zijn. Hierdoor ontstaat het probleem dat er op den duur te weinig ontwikkelaars zijn tijdens de ontwikkelfase. Een ander nadeel is als er slechts één product geproduceerd wordt. Als dit het geval is zal het rouleren van ontwikkeltaken nagenoeg onmogelijk zijn aangezien er in dat geval niet gewisseld wordt van project.

Tot slot is het nog mogelijk om de awareness van de gebruiker te verhogen. Een manier om dit te doen is om de code van de applicatie opensource aan te bieden aan de klant onder strikte voorwaarden. Dit stelt de klant in staat om zelf een security audit uit te (laten) voeren op het product, iets wat bij sommige projecten, zoals bij banken, soms zelfs verplicht is. Dit zorgt er voor dat de klant niet alleen afhankelijk is van de kwaliteiten van de producent maar deze kwaliteiten ook kan controleren. Daarnaast is het voor deze producent ook een extra controle op de geleverde producten.

Het nadeel van het opensource aanbieden van een product is dat dit niet altijd mogelijk is. In sommige gevallen verbieden derde partijen waarvan delen van de software zijn ingekocht (bijvoorbeeld JPEG compressie) het verspreiden van hun code als opensource. Daarnaast is de sourcecode makkelijker te stelen als deze opensource wordt aangeboden. Hiervoor kunnen echter strenge eisen en regels worden opgesteld, zodat dit risico wordt afgevangen.

7.2 Inrichting van het ontwikkelproces

Nadat we de awareness bij het management en de ontwikkelaars naar het wenselijke niveau hebben gebracht is het tijd om te kijken naar het ontwikkelproces. In de vorige sectie gaf ik al aan dat het rouleren van ontwikkeltaken kan bijdragen aan het verhogen van awareness bij ontwikkelaars. Het ontwikkelteam kan echter ook al eerder worden ondersteund in het bestrijden van het securiteitprobleem namelijk tijdens het ontwikkelproces zelf.

Zoals we in hoofdstuk 5 al zagen is het van belang om tijdens het ontwikkelproces de verantwoordelijkheid over de code vast te leggen. Dit heeft als direct resultaat dat ontwikkelaars zich verantwoordelijker zullen voelen over de code die ze produceren maar ook dat in het geval van een calamiteit de verantwoordelijke ontwikkelaar snel kan worden opgespoord zodat hij het probleem kan oplossen. Het voordeel hiervan is dat de ontwikkelaar de code waarin het probleem zich voordoet al kent, waardoor problemen sneller kunnen worden opgelost en het oplossen van het probleem ook minder foutgevoelig zal zijn.

Het vastleggen van deze verantwoordelijkheden kan op verschillende manieren gebeuren, zo kan er in de code zelf gewerkt worden met vaste commentaar opmaak waaraan te zien is welke ontwikkelaar verantwoordelijk is voor welk stuk code. Maar het is ook mogelijk om de code en verantwoordelijkheid te scheiden, door bijvoorbeeld een lijst bij te houden met coderegels en verantwoordelijke ontwikkelaars.

Een ander onderdeel van het inrichten van het ontwikkelproces is kennismanagement. Uit de interviews is gebleken dat binnen ontwikkelbedrijven kennis van onder andere security op verschillende wijzen en via verschillende kanalen wordt gedeeld. Bij kleine bedrijven is dit vaak nog te overzien en zal de kennis alle ontwikkelaars bereiken maar vooral bij de middelgrote bedrijven is het delen van kennis met alle ontwikkelaars een probleem. Dit kan echter vrij eenvoudig worden opgelost door één kanaal te kiezen waarin ontwikkelaars kennis kunnen vastleggen en delen. Eén van de bedrijven die meegedaan hebben met het interviewonderzoek maakte bijvoorbeeld gebruik van een Wiki waarmee ontwikkelaars kennis met elkaar deelden. Dit bleek zo nuttig dat deze Wiki ook gedeeltelijk werd opengesteld voor klanten die hiermee ook meer kennis kregen over de applicaties van de ontwikkelaar.

Om een dergelijk kanaal tot een succes te maken moet deze aan een aantal voorwaarden voldoen. Zo moet de kennis makkelijk te categoriseren zijn, moet er een mogelijkheid zijn om de gegevens te doorzoeken en moet de drempel om kennis toe te voegen laag zijn. Indien deze voorwaarden niet gehaald worden zal het voor de ontwikkelaars niet interessant zijn om hun kennis middels dit systeem te delen, wat er weer toe zal leiden dat de kennis nog steeds verdeeld wordt over verschillende (groepen) ontwikkelaars.

7.3 Tools ter ondersteuning van het ontwikkelproces

Het optimaliseren van het ontwikkelproces zoals beschreven in de vorige sectie is niet altijd nodig en/of wenselijk want dit proces kan ook op een andere manier worden ondersteund op het gebied van security namelijk door het gebruik van tools.

Voorbeelden van deze tools zijn bijvoorbeeld de Sourcecode Analyzer `Fortify` zoals behandeld in hoofdstuk 6, of tools als PMD of Findbugs zoals beschreven in hoofdstuk 5. Fortify is verkrijgbaar in command line versie of in een grafische versie met behulp van een Workbench. Het Fortify 360 pakket bevat drie verschillende scanners, een Source Code Analyzer voor gebruik tijdens de ontwikkelfase, een Program Trace analyzer voor gebruik tijdens de testfase en een Real-time Analyzer voor tijdens de productiefase van een applicatie.

Dit soort tools zijn uitermate geschikt om het ontwikkelproces aan te vullen door het gebruik van deze tools te implementeren en het gebruik hiervan verplicht te maken. Fortify heeft, zoals gebleken is uit het onderzoek, een complete oplossing in huis voor het analyseren van sourcecode.

Ook bij het onderzoek naar het gebruik van Fortify bij het scannen van PHP sourcecode is gebleken dat Fortify een goede aanvulling is op bestaande methoden zoals peer reviews. Het programma is in staat om in zeer korte tijd de code te analyseren en een zeer compleet overzicht van de gevonden bedreigingen te geven. Ontwikkelaars kunnen vervolgens deze bedreigingen nagaan en oplossen. Hiervoor biedt Fortify een gedetailleerde beschrijving bij elk gevonden probleem, compleet met mogelijke oplossingen.

Ontwikkelaars kunnen met behulp van dit soort tools eigen geschreven code valideren voordat deze wordt opgenomen in de uiteindelijke applicatie, of in een ‘code repository’ als SVN of CSV. Een van de voordelen van het gebruik van een repository is dat de versie van bestanden wordt bijgehouden. Hierdoor kan een ontwikkelaar precies zien wat de huidige versie van een bestand is en wat er gewijzigd is ten opzichte van de vorige versies. Een ander voordeel is dat vanuit de repository de gewijzigde versie kan worden geëxporteerd. Deze gewijzigde versie kan worden uitgerold over een of meerdere applicaties waardoor deze applicaties zullen beschikken uit de laatste versie van de gewijzigde bestanden uit de repository.

Het voordeel van het analyseren van de code voordat deze wordt opgenomen in de repository is dat er tijdens het uiteindelijke testtraject minder kans is op het vinden van security gerelateerde bugs, aangezien de ontwikkelaars zelf hun eigen code al hebben geanalyseerd. Daarnaast bleek uit het onderzoek dat een programma als Fortify niet alleen de fouten aangeeft in de code maar dat het de ontwikkelaar ook nog eens de informatie biedt om het probleem daadwerkelijk te kunnen begrijpen en ervoor te zorgen dat dezelfde fout voorkomen kan worden in de toekomst.

Hierbij moet wel worden opgemerkt dat vooral bij grote projecten met meerdere ontwikkelaars een tool als Fortify het beste individueel per ontwikkelaar kan worden ingezet. Op deze wijze kan de ontwikkelaar zijn eigen code scannen, waardoor hij beter in staat is om de resultaten van een scan te interpreteren en beter in staat is om de gevonden problemen op te lossen.

Een andere tool die nog niet is behandeld maar die ik zeker niet wil achterhouden is het gebruik van frameworks. Hoewel dit begrip voor de meeste ontwikkelaars niet nieuw is, zijn er toch weinig bedrijven waarbij het gebruik van een framework, al dan niet zelf geschreven, wordt toegepast. Hoewel het niet altijd handig is om een framework te gebruiken, zeker niet bij kleine applicaties, zijn er een aantal redenen met het oog op security waar het gebruik van een framework zeker voordelen biedt.

Het grootste voordeel van het gebruik van een framework is dat, mits goed geïmplementeerd, het framework een op zichzelf staand deel is binnen een applicatie. Alle functionaliteiten van de applicatie gaan lopen via het framework zoals onder andere de aansturing van de database, het controleren van user in- & output en system calls. Voordeel hiervan is dat het framework kan zorgen voor de controle van alle data, terwijl de applicatie alleen gebouwd hoeft te worden om de data te verwerken. Een ander bijkomend voordeel is dat het framework apart kan worden (door)ontwikkeld. Dit betekend dat als applicatie A en B gebruikmaken van hetzelfde framework en bij applicatie A wordt een security bug gevonden in het framework, het framework kan worden aangepast en opnieuw worden gedistribueerd over beide applicaties, waardoor niet alleen product A beter wordt maar ook product B.

7.4 Wat te doen na implementatie van een product

Niet alleen tijdens de ontwikkeling van een product is het mogelijk om processen te optimaliseren om zo een beter security beleid te creëren, ook na de implementatie is het zaak om middels een aantal goede processen er voor te zorgen dat eventuele (security) bugs snel kunnen worden opgelost.

Een van de middelen die hiervoor kan zorgen is een 'bug reposting tool' als bijvoorbeeld 'Bugzilla' of 'Mantis'. Deze web gebaseerde bug reporting tools bieden de klant de mogelijkheid om een bug te melden bij het ontwikkelbedrijf met de informatie die nodig is om het probleem te kunnen reproduceren en op te lossen. Hierbij kan gedacht worden aan informatie over het besturingssysteem, de browser of de versie van de applicatie. Het is uiteraard ook mogelijk dat het ontwikkelteam zelf de bugs invoert en kan dus ook gebruikt worden voor bugs die al tijdens het testtraject van de applicatie zijn gedetecteerd.

Gerapporteerde bugs kunnen vervolgens worden toegewezen aan een ontwikkelaar die dan weer een bepaalde status kan toewijzen aan de melding. Aan deze status kan vervolgens de klant weer zien dat er iets met de melding gebeurt en wat de huidige status is. Is de bug opgelost dan kan er een patch gegenereerd worden vanuit de Repository (zie vorige sectie) die vervolgens weer kan worden geïmplementeerd bij de klant.

Voordeel van het gebruik van een dergelijke manier van bug reporting is dat de versies uit de CVS gekoppeld worden aan bugs. Dit zorgt er namelijk voor dat een wijziging in de code te herleiden is naar een bug report en dus naar de ontwikkelaar die de code heeft aangepast. Hierdoor is de verantwoordelijkheid van de code duidelijk en is bij calamiteiten snel te vinden welke ontwikkelaar verantwoordelijk is voor welke code.

Er zijn ook nog andere voordelen in een dergelijke opzet die niet direct security gerelateerd zijn maar toch het vernoemen waard zijn. Zo zijn de eerder genoemde bug reporting tools in staat om gedetailleerde rapporten te genereren die gebruikt kunnen worden voor onder andere: het plannen van versie releases, het traceren van werkzaamheden per project, ontwikkelaar of klant en het indelen van problemen op prioriteit. Deze functionaliteiten kunnen bijdragen tot een verbetering in het ontwikkelproces en het motiveren van het management om dit soort tools in te zetten binnen het bedrijf.

8 Conclusie

Sinds het begin van deze eeuw zijn webapplicaties een steeds groter wordende trend binnen de IT-sector. Applicaties die via het internet beschikbaar worden gesteld aan gebruikers zonder dat hiervoor een programma geïnstalleerd hoeft te worden krijgen steeds vaker de voorkeur boven de klassieke desktopapplicatie. Maar het aanbieden van een programma via het internet brengt vooral op het gebied van security een paar grote gevaren met zich mee, gevaren die klassieke applicaties niet ondervonden.

Het 'Open Web Application Security Project' (OWASP) heeft in 2007 een top 10 opgesteld met de meest kwetsbare punten van webapplicaties waaronder: niet-gevalideerde input, fouten in de toegangscontrole en Cross-Site Scripting. Dit zijn typische problemen die webapplicaties met zich meebrengen doordat de applicaties in de browser worden uitgevoerd en de communicatie over het internet wordt opgebouwd. Dit zijn typische eigenschappen die webapplicaties onderscheiden van de klassieke desktopapplicaties.

Tijdens mijn literatuurstudie viel het me al op dat de aanwezige literatuur op het gebied van security veelal voor grote ontwikkelbedrijven bedoeld was, zoals beschreven door Kenneth Wyk [4]. Tijdens mijn werkzaamheden als ontwikkelaar is het me vaker opgevallen dat informatie voor kleine en middelgrote bedrijven verdeeld staat op het web maar niet terug te vinden is in de literatuur. Uit de interviews is me deels duidelijk geworden wat de oorzaak van dit probleem is: bij kleine en middelgrote bedrijven ontbreekt vaak een budget voor het volgen van cursussen en trainingen en zijn ontwikkelaars zelf verantwoordelijk voor het up-to-date houden van hun kennis.

Hier is als bedrijf goed op in te springen door zelf actief aan kennismanagement te doen door bijvoorbeeld het opzetten van een Wiki. Hierdoor wordt de communicatie tussen ontwikkelaars niet alleen bewaard voor andere ontwikkelaars maar kunnen ontwikkelaars ook elkaars bevindingen onderbouwen of aanvullen. Hierdoor biedt het bedrijf de ontwikkelaars de mogelijkheid om zelf actief bezig te zijn met hun kennis en elkaar onderling op te leiden.

Een ander probleem dat ik tijdens mijn onderzoek ben tegengekomen en waar ik vooraf niet aan had gedacht is de security awareness bij leidinggevenden en management. Ondanks dat securityproblemen in de IT-sector bijna dagelijks in het nieuws zijn, is het in het bedrijfsleven nog steeds een feit dat leidinggevenden dit probleem onderschatten. Uit de interviews en de literatuur blijkt dat dit probleem met name veroorzaakt wordt door het "security wordt pas een probleem als het een probleem is" syndroom. Hiermee bedoel ik dat de leidinggevenden van kleine en middelgrote bedrijven pas goed nadenken over securityproblemen nadat ze geconfronteerd worden met het misbruik van het probleem.

Een goed voorbeeld hiervan werd gegeven door een van de geïnterviewde ontwikkelaars die aangaf dat er pas tijd was om de security van de ontwikkelservers op orde te krijgen nadat de ontwikkelserver van een concurrerend bedrijf was gehackt. Dit is een serieus probleem, niet alleen omdat voor een goed security beleid een actieve aanpak vereist is, maar ook omdat blijkt dat ontwikkelaars in de praktijk de securityproblemen erkennen maar onvoldoende middelen krijgen om er daadwerkelijk iets aan te kunnen doen.

Een van die middelen die de ontwikkelaar bijvoorbeeld kan ondersteunen bij het controleren van zijn fouten tijdens het programmeren is een zogenaamde “Sourcecode Analyzer”, een programma dat sourcecode analyseert en onderzoekt op mogelijke security gerelateerde fouten. Uit het onderzoek bleek dat een tool als deze de ontwikkelaar niet alleen kan ondersteunen door de code op fouten te controleren, maar ook door de ontwikkelaar te leren wat hij fout heeft gedaan en hoe hij ervoor kan zorgen dat hij dit de volgende keer goed kan doen.

Het gebruik van de “Soursecode Analyzer” Fortify is sterk aan te raden voor ontwikkelaars van webapplicaties. Niet alleen is Fortify in staat om snel grote hoeveelheden broncode te analyseren, de resultaten van deze scan wordt vervolgens ook zeer gedetailleerd gepresenteerd aan de ontwikkelaar. Een applicatie als Fortify kan het beste worden ingezet door elke ontwikkelaar zijn eigen code te laten analyseren. Dit zorgt er niet alleen voor dat de ontwikkelaar goed in staat zal zijn om de resultaten van de scan te analyseren, maar ook zal hij sneller in staat zijn om de gevonden threats snel te kunnen verhelpen, aangezien hij bekend is met de code. Deze scans zouden een vast onderdeel moeten zijn van het ontwikkelproces. Een voorbeeld hiervan is bijvoorbeeld dat ontwikkelaars elke vrijdag middag verplicht de code moeten scannen die ze die week hebben geschreven.

Het scannen van de code zorgt er niet alleen voor dat de code wordt verbeterd, doordat Fortify bij elke gevonden bedreiging met een complete beschrijving van probleem en oplossing komt wordt de ontwikkelaar als het ware onderwezen op het gebied van security. Dit zorgt er niet alleen voor dat de ontwikkelaar in staat is om het gevonden probleem op te lossen, maar zorgt er ook voor dat de ontwikkelaar een zelfde probleem de volgende keer kan herkennen en vermijden.

Let wel, een sourcecode analyzer moet echt gezien worden als een ondersteunend middel. Ook uit voorgaand onderzoek [10] blijkt dat het nodig blijft om de code ook op andere manieren te controleren op fouten omdat een dergelijk programma niet alle fouten zal detecteren. Daarnaast is een programma als Fortify in staat om veel verschillende problemen te detecteren, het is echter aan de gebruiker om het programma zo in te stellen dat alleen die problemen worden weergegeven die echt relevant zijn voor de ontwikkelaar. Fortify bied hiervoor diverse middelen zoals filters en groepeer functies, deze stellen de ontwikkelaar in staat om op verschillende niveaus de resultaten van een scan weer te geven.

Om er voor te zorgen dat de ontwikkelaars de middelen krijgen die ze nodig hebben om de securityproblemen aan te kunnen pakken is het verstandig om te werken met een securitybudget. Ook al wordt er op dit moment binnen een bedrijf geen vast budget gebruikt voor security doeleinden, dan is het alsnog aan te raden om een schatting van de hoeveelheid security gerelateerde kosten te maken per project. Het doel hiervan is om security inzichtelijk te maken. Zoals ik al eerder aangaf wordt security bij veel bedrijven pas gezien als een serieus probleem als het al een probleem is. Wacht dit niet af, reserveer middelen om securityproblemen actief op te lossen en te bestrijden.

Tot slot wil ik benadrukken dat securityproblemen altijd zullen blijven bestaan, het zij in een andere vorm, het zij vanuit andere reden, er zal altijd rekening moeten worden gehouden met het feit dat er aanvallen kunnen plaatsvinden middels (misschien nu nog onbekende) exploits in een applicatie. Verwacht niet dat na het lezen van dit document alle security problemen op te lossen zijn. Ook al wordt ieder advies opgevolgd, als er niet wordt geanticipeerd op nieuwe ontwikkelingen in de toekomst is het gevecht tegen securityproblemen bij voorbaat al een verloren zaak. Zorg er dus voor dat het security beleid binnen uw onderneming niet alleen is gericht op het huidige security beeld maar houdt rekening met nieuwe ontwikkelingen die zich mogelijk in de toekomst kunnen voordoen en zorg er voor dat er bij nieuwe calamiteiten snel en adequaat kan worden ingegrepen.

8.1 Reflectie

Het onderzoek dat ik het afgelopen half jaar heb uitgevoerd is voor mijzelf een lastige opgave geweest. Door mijn dyslexie heb ik moeite met het lezen en schrijven van teksten, iets wat vooral tijdens de literatuurstudie en het schrijven van dit document voor de nodige problemen geeft gezorgd. Doordat ik vooraf volledig bewust was van het feit dat ik tegen deze problemen zou aanlopen kon ik tijdig maatregelen nemen om deze problemen tot een minimum te beperken. Een van de maatregelen is het vroegtijdig beginnen met het uitzoeken van de literatuur, zodat ik later niet in tijdsnood zou komen. Ook heb ik ervoor gezorgd dat al het geschreven werk door meerdere mensen is gecontroleerd om fouten in deze versie te voorkomen.

Achteraf gezien is de literatuurstudie me meegevallen, mede omdat ik al het nodige van dit onderwerp afwist door mijn vorige studies en werkervaring, maar ook door mijn interesse in het onderwerp heb ik de literatuur sneller kunnen bestuderen dan ik vooraf had gedacht. Hierdoor was ik in staat om meer literatuur te bestuderen dan ik vooraf had ingeschat.

Het onderzoek met behulp van het interviewen van de ontwikkelaars viel daar in tegen veel zwaarder uit dan ik vooraf had gedacht. Het afnemen van de interviews verliep mede dankzij de hulp van een aantal medestudenten vlot maar vooral tijdens het uitschrijven van de interviews en het verwerken van deze informatie bleek dit meer werk dan verwacht. Wat me ook is tegengevallen is het aantal reacties van ontwikkelaars op het verzoek om mee te werken aan het onderzoek. Dit had ik achteraf gezien beter anders aan kunnen pakken door in plaats van een mail te sturen de ontwikkelaars op een directere manier aan te spreken.

Het analyseren van de tool ‘Fortify’ is naar mijn mening goed gegaan. Helaas waren de testen, met behulp van de wizard en advanced opties van Fortify die ik vooraf had bedacht en besproken met mijn begeleider achteraf gezien niet allemaal mogelijk door beperkingen in Fortify. Dit is wel een op zichzelf staande conclusie, hoewel anders dan gehoopt. Uiteindelijk heb ik de tool naar mijn mening goed kunnen testen en met behulp van de al aanwezige conclusies van eerdere onderzoeken een conclusie weten te schrijven die ook voor bedrijven echt nuttig kan zijn.

Tijdens het schrijven van dit document heb ik mij niet strikt aan de planning gehouden. Ik ben mij er deels van bewust dat dit achteraf gezien beter had gekund, maar aan de andere kant heb ik er nu wel voor kunnen zorgen dat de dingen die ik onderzocht wilde hebben ook naar mijn mening goed onderzocht zijn. Uiteraard kunnen er altijd dingen nog beter worden gedaan en verder worden onderzocht maar ik ben van mening dat ik met de middelen die ik tot mijn beschikking had er alles aan gedaan heb om tot een zo goed mogelijk resultaat te komen.

8.2 Vervolgonderzoek

Tijdens mijn onderzoek zijn er een aantal aspecten van security tijdens het ontwikkelproces voorbij gekomen waar ik vanuit mijn onderzoek niets mee kon doen maar die wel nuttig of interessant zijn voor een eventueel vervolgonderzoek.

8.2.1 Browsersec

Een van deze onderwerpen is het 'Google Browser Security Handbook'[8]. Dit handboek probeert een samenvatting te geven van alle browser gerelateerde informatie op het gebied van security die nu her en der te vinden is op het internet. Voor ontwikkelaars kan dit handig zijn omdat webapplicaties die in een bepaalde browser draaien deels afhankelijk zijn van de security aspecten van die browser. Als een browser dus een bepaald securityprobleem heeft kan de ontwikkelaar hier bij de ontwikkeling van de applicatie rekening mee houden.

Het lijkt mij interessant om in dit handboek na te gaan en te kijken of wat er hier beschreven wordt volledig is (voor zover dat überhaupt kan) en of de informatie die beschreven staat ook correct is. Mocht blijken dat de informatie correct en volledig is dat is het logische vervolg in dit onderzoek om te bepalen wat dan de 'veiligste' browser is.

8.2.2 Secpoint

Om informatie te verzamelen voor mijn onderzoek heb ik een aantal security beurzen in het afgelopen jaar bezocht. Op één van die beurzen kwam ik de fabrikant 'Secpoint' tegen, een fabrikant van hardwarematige beveiligingsproducten. Eén van de producten die zij aanbieden is 'The Penetrator', een hardwarematige netwerk, systeem en applicatie penetrator tester. Doordat deze penetrator tester via het netwerk zijn werk doet is het ook mogelijk om webapplicaties te analyseren, hierbij kan gedacht worden aan websites, CMS-systemen maar ook bijvoorbeeld een applicatie voor internetbankieren.

Er kan onderzocht worden wat de meerwaarde van een hardwarematige tool als deze is op reeds bestaande softwarematige penetration tools en de correctheid van dit apparaat. Wat biedt een product als deze aan meerwaarde voor een bedrijf en wat zijn de mogelijkheden hiervan.

8.2.3 Frameworks

Hoewel ik het tijdens mijn scriptie al een paar keer heb genoemd ben ik niet in de gelegenheid geweest om de invloed van het gebruik van een framework op het securityaspect van een applicatie volledig te testen. Ik weet uit ervaring dat er bijvoorbeeld voor PHP een aantal frameworks zijn die, met het oog op security, een groot aantal verbeteringen bieden voor de applicaties die gebouwd worden op dit framework.

Welke frameworks zijn er voor ontwikkelaars van webapplicaties en welke middelen bieden zij de ontwikkelaar om securityproblemen aan te pakken? Welk framework is met het oog op security het beste en waarom? Wat bieden bestaande frameworks aan meerwaarde tegenover het zelf bouwen van een framework?

- [8] http://code.google.com/p/browsersec/wiki/Main

9 Bibliografie

[1] Gary McGraw, *Software Security - Building security in*. Boston: Addison Wesley, 2006.

[2] Roger S. Pressman & David Lowe, *Web engineering*. New York: Mc Graw Hill, 2009.

[3] Michael Howard & Steve Lipner, *The security development lyfecycle*. Redmond: Microsoft Press, 2006.

[4] Kenneth R. van Wyk, "Secure Development Processes," , Leuven, 2008.

[5] The Open Web Application Security Project. (2009, Maart) The Open Web Application Security Project. [Online]. http://www.owasp.org

[6] ILia Alshanetsky, *php|architec's Guide to PHP Security*. Toronto: Marco Tabini & Associates, inc, 2005.

[7] Chris Snyder & Michael Southwell, *Pro PHP security*. New York: Apress, 2005.

[8] Chris Shiflett, *Essential PHP Security*. sebastopol: O'Reilly MEdia, inc, 2006.

[9] The PHP group. (2009, Maart) PHP: Hypertext Preprocessor. [Online]. http://www.php.net/

[10] Ware & Fox, "Securing Java Code: Heuristics and An Evaluation of Static Analysis Tools," in *Securing Java Code: Heuristics and An Evaluation of Static Analysis Tools*, Harrisonburg, 2008, pp. 12 - 21.

[11] Pugh Ayewah, "A report on a survey and study of static analysis users," in *A report on a survey and study of static analysis users*, Maryland, 2008, pp. 1 - 5.

[12] E.G.M. Strijbos, "Web Application Security Certificates," in *Master Thesis Radboud Universiteit*, Nijmegen, 2008.

[13] MKB Nederland. (2009, Januari) MKB Nederland. [Online]. http://www.mkb.nl/

[14] Gwen Bell (Computer History Museum), Jan Zimmerman, Jacqueline Boas, Bill Boas Dag Spicer. Computer History Museum. [Online]. http://www.computerhistory.org/internet_history/

[15] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Web_application

[16] Jesse James Garrett. (2005, Jan.) adaptivepath. [Online]. http://www.adaptivepath.com/ideas/essays/archives/000385.php

[17] Wikipedia. [Online]. http://en.wikipedia.org/wiki/Web_2.0

[18] Tim O'Reilly. (2006, Oct.) Orielly. [Online]. http://radar.oreilly.com/archives/2006/12/web-20-compact.html

[19] Kelly Jackson Higgins. (2006, Nov.) Dark Reading. [Online].
http://www.darkreading.com/document.asp?doc_id=111482&f_src=darkreading_section_296

[20] Jeremiah Grossman. (2006, Nov.) jeremiahgrossman.blogspot.com. [Online].
http://jeremiahgrossman.blogspot.com/2006/11/web-application-security-risk-report.html

[21] Larry Greenemeier. (2006, Oct.) informationweek. [Online].
http://www.informationweek.com/blog/main/archives/2006/11/etailers_leavin.html

[22] Jeff Atwood. (2008, Dec.) Coding Horror. [Online].
http://www.codinghorror.com/blog/archives/001198.html

10 Bijlagen

10.1 Interview

Structuur: Open / gestuurd

Topic list:

- Achtergrond ontwikkelaar
- Achtergrond bedrijf / afdeling
- Ontwikkelproces
- Toekomstbeeld

Vragen:

- **Achtergrond ontwikkelaar**
  - Opleidingsniveau, werkervaring & functie(s)
  - Ervaring PHP (en/of andere webprogrammeertalen) in samenwerking met beveiliging
  - Wat is het beveiligingsprobleem (gerelateerd aan webapplicaties)

- **Achtergrond bedrijf / afdeling**
  - Hoe groot is het bedrijf
  - Wat zijn de webapplicaties/web gerelateerde producten van het bedrijf
  - Waarin verschillen de applicaties/producten onderling
  - Is er een budget voor ontwikkeling en specifiek voor beveiliging
  - Wordt er gebruik gemaakt van testen en/of analyse software voor het detecteren van beveiligingslekken

- **Ontwikkelproces**
  - Is er een vast proces
    - Zo ja: wat is dit proces
    - Zo nee: waarom niet
  - Is er nagedacht over beveiligingswaarborging
  - Wie is er binnen het bedrijf / de afdeling / het project verantwoordelijk voor de correctheid / veiligheid van de code
  - Hoe wordt er omgegaan met beveiligingslekken (melden, feature- v.s. bugrequests, afhandeling)
  - Welke bronnen worden er binnen het bedrijf gebruikt om kennis over beveiliging bij te houden.

- **Toekomstbeeld**
  - Hoe wordt ervoor gezorgd dat het kennisniveau m.b.t. beveiliging op peil blijft
  - Wat verwacht je qua ontwikkelingen in de markt
  - Wat zie je zelf als verbeterpunten voor jezelf en/of het bedrijf