Interview

**Julien Rentrop en Peter Schuler hebben voor Ordina cursussen over Java web application security gegeven, waaruit een grote belangstelling voor het onderwerp bleek. De cursus startte in 2010, dus ruim voor het Diginotar-debâcle. Inmiddels lijkt security een echte hype te zijn.Tijdens J-Fall gaven Rentrop en Schuler een boeiende presentatie over security.**

# Security = Awareness

## Veiligheid schiet op veel websites ernstig tekort

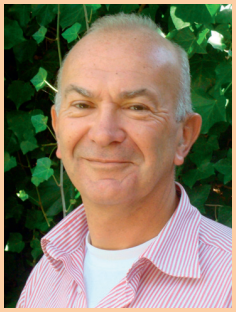
Nu lift het onderwerp dus mee op de media-hype over beveiliging van webapplicaties, maar het zou geen hype mogen zijn. Als ontwikkelaars goed met dit onderwerp omgaan, zou er zelfs helemaal geen hype zijn geweest.

Bij security draait alles om 'awareness', dag in dag uit. De veiligheid van een applicatie zou in de genen van iedere ontwikkelaar en software engineer moeten zitten. Het securityprobleem bestaat eigenlijk al zo lang het internet bestaat, maar nog steeds gaat het regelmatig fout. Om fouten te voorkomen moet je in eerste instantie gedisciplineerd werken, stelt Peter Schuler. "Er zijn veel regels. Bij de parameters moet je bijvoorbeeld string concatenatie gebruiken als je naar de database gaat. En zijn er meer regels, die zo'n beetje in steen vastliggen. Maar toch zijn er nog veel mensen die daarvan afwijken omdat ze niet beseffen waarom die regels er zijn".
"Ik heb het meegemaakt dat mensen niet weten hoe een lijstje van waardes in een JPQL query wordt gemaakt en dan maar een for-lusje schrijven en de waardes er met de hand in plakken. Dat werkt ook, maar er is een goede parameter voor die je kunt gebruiken. Omdat mensen niet beseffen dat het gevaarlijk is wat ze doen, kiezen ze vaak de makkelijkste weg. Sinds ik deze cursus geef vind ik dagelijks sites die ik zelf gebruik, waar ik gaten in vind. Dat geeft aan dat er nog veel werk moet worden verzet", aldus Peter.
Julien voegt toe: "Goede inhoudelijke kennis van de nieuwe technologie en frameworks binnen Java is erg belangrijk om securitygaten te voorkomen. De techniek ontwikkelt zich natuurlijk steeds verder. Als je met een nieuw project aan de gang gaat moet je je dus ook verdiepen in de daarbij gebruikte technologie en kijken welke mechanismes er zijn om de veiligheid te waarborgen. Wacht ook niet tot een security-review voorbij komt, maar zorg dat je dat vóór bent. En als er dan een gat aan het licht komt, bekijk dat goed en bestudeer hoe je dat in de toekomst kunt voorkomen".
Tijdsdruk is een belangrijke factor bij het ontstaan van veiligheidslekken. Een andere is het doorbouwen op een ouder, bestaand systeem. Dan moet voldoende aandacht worden besteed aan de bestaande software om te bezien of daarin de security goed is geregeld. Je moet dit uiteraard ook in kaart brengen voor je opdrachtgever.

### Waterval

Zowel in waterval ontwikkeling als bij agile methodieken liggen veiligheidsrisico's op de loer. "De methode is niet echt van invloed. Er zijn altijd veel

Huub Jansen, directeur Ordina J-Technologies, Julien Rentrop en Peter Schuler tijdens J-Fall in Nijkerk.



**Robert de Ruiter**
communicatietrainer.

lekken geweest. Maar agile leent zich ervoor dat je snel gaat werken. Dat je nog snel even iets extra's bouwt om de klant tevreden te stellen. Je moet je dus goed realiseren dat veiligheid iets is van het hele multidisciplinaire team. Ook de tester moet zich bewust zijn van het veiligheidsaspect. Security is een belangrijke niet-functionele requirement, maar ook een waar je niets van merkt. Als hier iets niet goed zit merkt niemand er iets van, tot het mis gaat. Het gemene is dat je als programmeur duizend fouten kunt maken, maar de hacker hoeft er maar één te vinden", zegt Peter. Uiteindelijk denkt hij wel dat agile beter is dan waterval, mits het zorgvuldig wordt uitgevoerd.

Garanties kun je echter nooit geven. Zelfs de grootste beveiligingsexperts hanteren disclaimers, waarin ze iedere aansprakelijkheid voor security-breaks afwijzen. Het is wel zo dat een goed begrip van security-aspecten de software op dit punt behoorlijk kan verbeteren. Julien en Peter gebruiken voor hun cursus veel informatie van de OWASP, het Open Web Application Security Project. Dit verzamelt informatie en stelt onder andere een Top-10 van grootste beveiligingslekken samen met als doel om veiliger applicaties te kunnen ontwikkelen. Maar ook dit geeft geen garantie voor volledig veilige software, want er zijn vele tientallen aanvallen te bedenken. De in de Top-10 genoemde risico's zijn wel de meest voorkomende bij softwareontwikkelaars en kunnen veel schade aanrichten.

**De resultaten in een kluis, maar met een kabeltje naar buiten.**

# Je apps beveiligen

Hoe kan je je eigen applicaties verbeteren. Een aantal tips van Peter Schuler en Julien Rentrop:

### Injection:

Bedenk goed dat injection niet alleen SQL maar ook JPA applicaties kan raken. De volgende JPA query is onveilig:

```
String jpql = "select p from Persoon p where emailadres
= '" + emailadres + "'";
Query query = em.createQuery(jpql);
query.excecute();
```

Door user input door string concatenatie is deze applicatie gevoelig voor JP-QL injection. Beter is het gebruik van named parameters:

```
String jpql = "select p from Persoon p where emailadres
= :emailadres";
Query query = em.createQuery(jpql);
query.setParameter("email", emailadres);
query.excecute();
```

Voor iedere mogelijke variabele is er een setParameter() functie. Gebruik deze dus altijd! Ook de JPA Criteria API is veilig in gebruik.

Het bovenstaande voorbeeld laat zien dat Injection overal kan voorkomen. Niet alleen in database communicatie maar in alle communicatie die via String tot stand komt. Denk hierbij aan LDAP, XPath, Shell commands, Lucene Queries en dynamsche code evaluatie.

Ook NoSQL systemen als CouchDB4J, Cassandra en Neo4j worden door Strings aangestuurd.

### Cross Site Scripting (XSS)

Bij Cross Site Scripting (XSS) word HTML/Javascript gebruikt op plekken waar normaal dynamische content wordt getoond. Denk bijvoorbeeld aan namen, een blog comment, een foutmelding of zelfs een URL naar een plaatje. Als een aanvaller in plaats van normale tekstuele input HTML/Javascript opstuurt dan kan hij de site aanpassen en overnemen.

Als je bijvoorbeeld bij een blog comment niet "Jan" invult maar "<script>alert("XSS Attack!");</script> " zal iedere volgende bezoeker een Javascript pop-up krijgen. Om dit te voorkomen passen we escaping toe. Escaping is het omzetten van verstorende tekens naar valide tekens. Voor HTML betekent dat < > worden omgezet naar > en <. Hierdoor wordt het stuk tussen de < > niet meer in de HTML opgenomen, maar gewoon als tekst gezien.

Een XSS gat ontstaat bijvoorbeeld als volgt:

```
<input type="text" name="naam" value="${param.naam}" />
```

Deze regel JSP/Expression Language vult een HTML formulier meteen met de input. Hoeft de gebruiker niet alles opnieuw in te vullen als het formulier niet door de validatie komt. Maar er wordt nu wel dynamische user input rechtstreeks in de HTML geplaatst.

Beter is de volgende oplossing:

```
<input type="text" name="naam" value="<c:out
value="${param.naam}">" />
```

De Standard tag library <c:out> zorgt ervoor dat alle gevaarlijke characters netjes worden omgezet.

"Bovendien kunnen securitygaten heel diep zitten, waar je als ontwikkelaar nauwelijks bij komt. Als het een defect is in de Java runtime omgeving bijvoorbeeld. Ook zitten security en useability elkaar vaak in de weg. Zo is er een discussie rond de ING Bank. Die heeft een handige feature dat als je een fout rekeningnummer intikt, de naam van de persoon, die bij dat nummer hoort naar voren komt. Je zou dan kunnen zeggen dat het een lek is, omdat je zo veel namen kunt achterhalen. Maar het is ook heel handig. Bij DigiD moesten de resultaten in een afgesloten kluis worden bewaakt. Maar een systeembeheerder vond het handig om een kabeltje naar buiten te trekken en de gegevens in een Windows domein te hangen. Op papier was alles beveiligd, maar in de praktijk werd gekozen voor handiger werken", signaleert Peter.

## Injection

Een van de grootste risico's is injection. Meestal wordt gesproken over SQL injection, maar het is eigenlijk altijd mogelijk als je een string based commandosysteem hebt, waarbij je rechtstreeks data die je niet kunt vertrouwen toevoegt aan de commando's. Dat kan dus ook als je XML-bestanden doorloopt met Xpath. Of als je commando's uitvoert op de doelcomputer. Er zijn voorbeelden van injection waarbij een Shell commando werd gestart met PHP, maar je kon dat Shell commando aanpassen met chmod en iedereen schrijfrechten geven… «

Alle moderne web frameworks hebben escapen tegenwoordig standaard aan staan. Zet dit dus ook nooit uit. Verder moet je goed opletten in welke context de dynamische content wordt gebruikt. Javascript, URL, CSS en HTML hebben allemaal verschillende escaping regels.

Een goede start is de OWASP XSS Prevention Cheat Sheet: http://tinyurl.com/7ky8d7

Als je wilt weten wat er allemaal mis kan gaan als je de gebruiker blootstelt aan een XSS aanval moet je kijken naar de mogelijkheden van de BeEF tool. Deze injecteert een stukje javascript via XSS waardoor de browser een zombie wordt die via active pulling commando´s ophaalt bij een centrale server. Via HTML5 Cross Origin Requests (COR) kan deze tool zelfs het interne (!) netwerk scannen.

XSS slaat zo een brug tussen het internet en het intranet.

## Session Hijacking

Session cookies worden gebruikt door Applicatie Servers en browsers om gebruikers te identificeren en te authenticeren. Session Hijacking is het stelen van een session cookie om zo de sessie van een ingelogde gebruiker over te nemen.

Populaire methoden zijn netwerk sniffing en via een XSS aanval met Javascript het cookie stelen. Om dit te voorkomen kent de Servlet 3.0 een aantal nieuwe opties om Session Cookies te beveiligen.

Dit kan door de onderstaande code in de web.xml op te nemen:

```
<session-config>
        <session-timeout> time-out -in-minutes </
session-timeout>
        <cookie-config>
                <http-only>true</http-only>
                <secure>true</secure>
        </cookie-config>
  </session-config>
```

De opties in de cookie-config geven aan dat het cookie niet mag worden benaderd via Javascript (http-only) en alleen over een HTTPS verbinding (secure).

In de Servlet 3.0 spec kan je tegenwoordig ook URL rewriting uitschakelen en heb je de mogelijkheid om de naam van je session cookie te kiezen in plaats van JSESSIONID. Kijk hiervoor naar de documentatie. Let wel op bij het hernoemen van je session cookie dat session aware load balancers ook deze wijziging moeten meekrijgen.

Peter Schuler werkt als zelfstandig Java Architect en Trainer.

Julien Rentrop, Java software engineer bij Ordina.

**De BeEF tool injecteert een stukje javascript via XSS.**

**Referenties:**
**OWASP:**
https://www.owasp.org/index.php/Main_Page
**BeEFF-tool:**
http://beefproject.com/.