GOVCERT.NL

Factsheet FS 2011-08

Checklist beveiliging webapplicaties

De beveiliging van webapplicaties staat de laatste maanden sterk in de belangstelling. Diverse incidenten op dit gebied tonen aan dat webapplicaties nog vaak triviale kwetsbaarheden bevatten die de ontwikkelaar eenvoudig had kunnen voorkomen. De toename van het aantal incidenten en de aandacht hiervoor in de media leidt bij veel eigenaren en beheerders van webapplicaties tot de vraag: hoe (on)veilig is mijn webapplicatie nu eigenlijk?

Dit factsheet beschrijft hoe u een basiscontrole kunt uitvoeren op de veiligheid van een webapplicatie zodat u snel inzicht krijgt in de aanwezigheid van veel voorkomende kwetsbaarheden.

De belangrijkste feiten op een rij:

- Veel webapplicaties bevatten kwetsbaarheden die eenvoudig voorkomen hadden kunnen worden.
- Veel kwetsbaarheden worden veroorzaakt door één of meerdere van deze fouten:
  1. Software is niet voorzien van de laatste patches.
  2. De webapplicatie filtert de invoer van gebruikers niet of niet goed.
  3. De webapplicatie codeert gevaarlijke karakters in de uitvoer niet of niet goed.
  4. De webapplicatie heeft een onveilig sessiemechanisme.
  5. De webapplicatie maakt op een onveilige manier gebruik van een database.
- Controle van uw webapplicatie op bovenstaande fouten geeft u een eerste inzicht in de beveiliging van deze webapplicatie.

Achtergrond

GOVCERT.NL biedt op zijn website (www.govcert.nl) het whitepaper 'Raamwerk Beveiliging Webapplicaties'[1] aan. Daarin staat beschreven welke kwetsbaarheden zich vaak in webapplicaties bevinden en hoe ontwikkelaars en andere betrokkenen deze kunnen voorkomen. Het whitepaper behandelt een uitgebreid scala aan kwetsbaarheden en maatregelen bij webapplicaties, zowel in de breedte als in de diepte. Het besteedt niet alleen aandacht aan de webapplicatie zelf, maar ook aan aangrenzende zaken zoals de beveiliging van onderliggende besturingssystemen en de veilige opbouw van de infrastructuur. Dit factsheet bouwt voort op het Raamwerk Beveiliging Webapplicaties.

Aanpak

Dit factsheet geeft een handvat om de veiligheid van een webapplicatie op vijf essentiële punten te beoordelen. Deze punten zijn de oorzaak van veel kwetsbaarheden. Door steekproefsgewijs de webapplicatie te beoordelen op deze kernpunten ontstaat inzicht in de sterke en zwakke punten van een webapplicatie. Op basis hiervan kunt u een diepgaande controle uitvoeren en veel voorkomende kwetsbaarheden zo snel mogelijk verhelpen.

Bij veel webapplicaties is het niet mogelijk om bovenstaande vijf kernpunten op alle plekken te controleren. Selecteer daarom de meest dynamische onderdelen van de webapplicatie (die met veel gebruikersinteractie) en controleer deze. Voor de hand liggende onderdelen zijn invulformulieren, zoekfuncties en inlogschermen.

Kernpunten

De checklist voor de beveiliging van webapplicaties bestaat uit vijf kernpunten: patching, invoervalidatie, uitvoervalidatie, sessieonderhoud en databasegebruik. Op de volgende pagina staat een overzicht van deze kernpunten met daarbij de controlehandelingen om de beveiliging van uw webapplicatie te toetsen. Daarnaast bevat het overzicht verwijzingen naar meer achtergrondinformatie. Wanneer u in de tekst een verwijzing als ➲102 tegenkomt, betekent dit dat u meer informatie over dit onderwerp kunt terugvinden op pagina 102 van het whitepaper ‘Raamwerk Beveiliging Webapplicaties’.

[1] Zie http://www.govcert.nl/dienstverlening/Kennis+en+publicaties/whitepapers/raamwerk-beveiliging-webapplicaties.html

Wilhelmina van Pruisenweg 104
2595 AN Den Haag

Postbus 117
2501 CC Den Haag

T 070 888 7 555
E info@govcert.nl
I www.govcert.nl

De vijf kernpunten voor webapplicatiebeveiliging

1. **Alle systemen zijn voorzien van de laatste updates ➲18, ➲51, ➲108**
   - Zijn de laatste patches van besturingssystemen, webservers, applicatieservers, applicaties en bibliotheken geïnstalleerd?
     Voorbeeld: de laatste versies van Apache, OpenSSL, PHP en Joomla zijn geïnstalleerd.
   - Zijn workarounds voor ernstige kwetsbaarheden doorgevoerd in de gevallen dat er geen patch van de leverancier beschikbaar is?
   - Is een proces ingericht om te monitoren op nieuwe kwetsbaarheden?
   - Is een proces ingericht om de ernst van nieuwe kwetsbaarheden in te schatten en passende maatregelen (patches, workarounds) door te voeren?

2. **De webapplicatie controleert en filtert alle invoer van gebruikers ➲86, ➲92, ➲116**
   - Staat de webapplicatie alleen invoer van gebruikers toe wanneer deze voldoet aan een vooropgesteld patroon? ➲93
     Voorbeeld: het patroon <naam>@<domein>.<tld> met een maximale lengte van 25+25+5 (naam+domein+tld) karakters voor een emailadres
   - Blokkeert de webapplicatie potentieel gevaarlijke sleutelwoorden en karakters in de gebruikersinvoer op basis van een blacklist? ➲87, ➲93, ➲115
     Voorbeeld: het 'DROP TABLE' sleutelwoord of het pipe-teken ('|')
   - Maakt de webapplicatie potentieel gevaarlijke karakters zoals de apostrof onschadelijk door deze te voorzien van een escape ('\')? ➲94
     Voorbeeld: het omzetten van de invoer ''s-Gravenhage' in '\'s-Gravenhage'
   - Normaliseert de webapplicatie de gebruikersinvoer voor deze te controleren? ➲97, ➲118
     Voorbeeld: het normaliseren van het pad "/img/path/../../script.aspx" naar "/script.aspx"
   - Vindt de uiteindelijke validatie van gebruikersinvoer plaats aan de serverzijde? ➲100
   - Controleert de webapplicatie alle dynamische invoer? Dus niet alleen de velden die de gebruiker expliciet invoert, maar ook de inhoud van te manipuleren invoer als URL's, cookies en HTTP-headers. ➲92

3. **De webapplicatie codeert dynamische uitvoer ➲97**
   - Codeert de webapplicatie potentieel gevaarlijke karakters in de dynamische onderdelen van de uitvoer (zoals het '<'-teken)? ➲97
   - Houdt de webapplicatie bij het coderen rekening met de plek in de uitvoer waar deze potentieel gevaarlijke karakters terugkeren? Uitvoer bestaat veelal uit headers, scripts, HTML en andere onderdelen. HTML-codering verschilt hierbij bijvoorbeeld van URL-codering. ➲98
     Voorbeeld: het kleiner-dan-teken ('<') moet HTML-gecodeerd worden als '<' maar URL-gecodeerd worden als "%3E"

4. **De webapplicatie onderhoudt een veilige sessie met de gebruiker**
   - Koppelt de webapplicatie de inhoud van een sessie-variabele aan een IP-adres? ➲95, ➲119
   - Breekt de webapplicatie een sessie automatisch af na een bepaalde tijd van inactiviteit? ➲131, ➲142
   - Biedt de webapplicatie de gebruiker de mogelijkheid de sessie zelf te beëindigen? ➲142
     Voorbeeld: het opnemen van een logout-functie in de pagina
   - Zijn formulieren voorzien van een dynamisch token? ➲95

5. **De webapplicatie maakt op een veilige manier gebruik van een database**
   - Maakt de webapplicatie voor het aanroepen van databases gebruik van geparameteriseerde queries in plaats van dynamische strings? ➲98

     ✘*Fout voorbeeld:*
     ```
     $sth = $dbh->prepare("SELECT * FROM products WHERE id = " . $productid);
     $sth->execute();
     ```
     ✔ *Goed voorbeeld:*
     ```
     $sth = $dbh->prepare("SELECT * FROM products WHERE id = ?");
     $sth->execute($productid);
     ```
   - Heeft het account waarmee de webapplicatie een verbinding opzet met de database beperkte rechten op deze database? ➲49, ➲92, ➲102
   - Versleutelt de webapplicatie gevoelige informatie in de database? ➲148
   - Toont de webapplicatie eventuele foutmeldingen niet aan de gebruiker? ➲65-67, ➲72
   - Zijn bovenstaande principes ook toegepast op *stored procedures*? ➲100

GOVCERT.NL is het Cyber Security & Incident Response Team van de Nederlandse overheid. Vragen over de inhoud van deze factsheet kunt u richten aan info@govcert.nl of telefonisch via 070-8887555.

Wilhelmina van Pruisenweg 104
2595 AN Den Haag

Postbus 117
2501 CC Den Haag

T 070 888 7 555
E info@govcert.nl
I www.govcert.nl

Controleren op de kernpunten
Per kernpunt is hierna een lijst aan controles opgenomen die u kunt uitvoeren op uw applicatie. Er bestaan diverse mogelijkheden om deze controles uit te voeren. Als u beschikt over de broncode van de webapplicatie heeft het de voorkeur om voor een whitebox-aanpak te kiezen: u bekijkt bepaalde onderdelen van de code om te zien hoe deze omgaat met bijvoorbeeld in- en uitvoer.

Als een whitebox-aanpak niet tot de mogelijkheden behoort, kunt u kiezen voor een grey- of blackbox-aanpak. U beschouwt de webapplicatie als een 'zwarte doos' waaraan u allerlei verzoeken toestuurt. Door deze verzoeken te manipuleren en het resultaat ervan te evalueren, ontstaat inzicht in de wijze waarop de webapplicatie deze verzoeken afhandelt. Hoewel het mogelijk is om een aantal zaken te controleren op basis van alleen een webbrowser, kunt u beter specialistische tooling gebruiken zoals *reverse proxies* en op die manier verzoeken op laag niveau manipuleren.

Let op: de kernpunten richten zich alleen op de meest voorkomende kwetsbaarheden en de meest voorkomende varianten hiervan. Als na de controles blijkt dat uw webapplicatie niet kwetsbaar is voor de lekken, betekent dit niet automatisch dat uw webapplicatie geen kwetsbaarheden bevat. Zie het whitepaper voor een beschrijving van andere kwetsbaarheden in webapplicaties.

Kernpunt #1: patching
Webapplicaties zijn vaak afhankelijk van andere applicaties. Deze applicaties kunnen elk afzonderlijk ook weer kwetsbaarheden bevatten. Om te voorkomen dat kwaadwillenden de webapplicatie via dergelijke kwetsbare applicaties aanvallen, is het belangrijk dat bekende kwetsbaarheden worden verholpen via patches of workarounds. Aan de basis hiervan staat een goed ingericht patchmanagementproces: de organisatie monitort hierbij continu op nieuwe kwetsbaarheden, beoordeelt deze en voert wijzigingen door op basis van deze beoordeling. Vergeet niet om ook patches te installeren voor webservers (bijvoorbeeld Apache), applicatieservers (bijvoorbeeld WebSphere), content management systemen (bijvoorbeeld Joomla) en bibliotheken (bijvoorbeeld OpenSSL).

Controles:
- breng de versies van gebruikte software in kaart
- controleer of deze versies up-to-date zijn (geen bekende kwetsbaarheden bevatten) via de websites van leveranciers
- controleer of uw software onopgeloste kwetsbaarheden bevat

Kernpunt #2: afhandeling van gebruikersinvoer
Vrijwel elke webapplicatie maakt tegenwoordig gebruik van invoer door de gebruiker. Gebruikers zijn dus in theorie in staat om de werking van de webapplicatie te beïnvloeden via invoer. Malafide invoer maakt aanvallen als SQL-injectie (➲36) en Cross-Site Scripting (XSS) (➲65) mogelijk. Om dit soort aanvallen te voorkomen moet de webapplicatie in eerste instantie alle invoer beoordelen op basis van de vastgelegde te verwachten invoer (een '*whitelist*'). Zo zou een telefoonnummer bijvoorbeeld alleen cijfers mogen bevatten en zou een postcode moeten bestaan uit vier cijfers gevolgd door twee letters (➲93). Daarna moet de webapplicatie ook een controle uitvoeren op potentieel gevaarlijke sleutelwoorden in de invoer op basis van een *blacklist*. Een voorbeeld van een gevaarlijk sleutelwoord is 'DROP' bij een SQL-statement.

Tot slot moet de webapplicatie karakters die de syntax van een aanroep kunnen verstoren, onschadelijk maken door deze te *'escapen'*. Denk hierbij aan de apostrof ('') bij het samenstellen van queries en het pipe-teken ('|') bij het samenstellen van commando's op het niveau van het besturingssysteem. Belangrijk bij dit alles is dat de webapplicatie alle gebruikersinvoer eerst normaliseert: de webapplicatie verwijdert alle coderingen, NULL-karakters, et cetera. Voert de webapplicatie deze normalisatie niet uit, dan kan een aanvaller filters mogelijk omzeilen door malafide invoer te verbergen. Ook moet de webapplicatie alle mogelijke invoer controleren en niet alleen de invoer die de gebruiker via de browser kan beïnvloeden. Een aanvaller kan met zijn eigen tools namelijk alle onderdelen van een verzoek aanpassen.

Wilhelmina van Pruisenweg 104
2595 AN Den Haag

Postbus 117
2501 CC Den Haag

T 070 888 7 555
E info@govcert.nl
I www.govcert.nl

Controles:
- controleer of de webapplicatie afwijkende invoer afwijst; controleer hierbij in ieder geval op de verwerking van karakters als '<', '>', '", '"', ‘’ en '|'
- controleer of de controles ook plaatsvinden op minder voor de hand liggende invoermogelijkheden zoals cookiewaarden en HTTP-headers

Kernpunt #3: afhandeling van dynamische uitvoer
De invoer die een gebruiker verstuurt, gebruikt de webapplicatie in veel gevallen direct (via reflectie) of indirect (via tussenopslag) in de uitvoer. Een voorbeeld van direct gebruik is wanneer de webapplicatie de originele zoekterm in de uitvoer van een zoekfunctie weergeeft (bijvoorbeeld “resultaten voor '<zoekterm>'”). Een voorbeeld van indirect gebruik is het weergeven van gebruikerscommentaren bij blogpostings. Het niet correct coderen van gevaarlijke karakters in deze uitvoer kan leiden tot problemen als Cross-Site Scripting (XSS). De webapplicatie moet daarom gevaarlijke karakters in de uitvoer coderen wanneer deze is gebaseerd op de invoer van de gebruiker. Zo moet een webapplicatie een kleiner-dan-teken ('<') afkomstig uit de invoer HTML-coderen als '<' (➲**98**) wanneer dit teken wordt getoond in de HTML-code.

Controles:
- bekijk of invoer van de gebruiker terugkeert in de uitvoer
- controleer of de webapplicatie deze invoer juist codeert; controleer hierbij in ieder geval op karakters als '<' en '>'

Kernpunt #4: sessieonderhoud
Vrijwel elke webapplicatie bouwt een sessie op met de gebruiker. In veel gevallen komt deze sessie tot stand door het plaatsen van een cookie op de pc van de gebruiker met daarin een uniek kenmerk (sessie-ID). Belangrijk is dat de webapplicatie een sessie-ID koppelt aan een IP-adres. Hierdoor voorkomt de webapplicatie dat een malafide gebruiker de inhoud van een cookie kopieert en zelfstandig een verbinding met de webapplicatie opzet op basis van het gestolen sessie-ID. Verder moet de webapplicatie een sessie niet oneindig lang actief houden. Na een bepaalde periode van inactiviteit (bijvoorbeeld 10 minuten), moet de webapplicatie de sessie afbreken. Hierdoor zijn sessie-ID’s slechts beperkt houdbaar en heeft een aanvaller slechts zeer beperkt de tijd om misbruik te maken van een gestolen sessie-ID. Naast het feit dat de webapplicatie een sessie ‘onklaar’ kan maken, moet ook de gebruiker zelf de mogelijkheid worden geboden om een sessie te beëindigen. Een voorbeeld hiervan is de mogelijkheid tot een *logout* op websites.

Controles:
- controleer of het mogelijk is om eenzelfde sessie-ID vanaf verschillende IP-adressen te gebruiken
- controleer of een sessie na een bepaalde timeout niet meer beschikbaar is
- bekijk of de website de mogelijkheid biedt de sessie te beëindigen

Kernpunt #5: databasegebruik
Veel webapplicaties maken voor de opslag van gegevens gebruik van databases. Deze databases zijn gewilde prooien voor kwaadwillenden want daarin staat over het algemeen de meest interessante informatie. Absoluut essentieel bij het gebruik van databases is dat de webapplicatie gebruik maakt van geparameteriseerde queries voor het benaderen van de database. Dit betekent dat de webapplicatie het SQL-statement niet dynamisch samenstelt door verschillende strings 'aan elkaar te plakken' maar dat dynamische invoer van de gebruiker terecht komt op de plek van zogenoemde *placeholders* (➲**100**) in de statische query (zie voor een voorbeeld de checklist op pagina 2). Door gebruik te maken van placeholders voorkomt de ontwikkelaar al veel van de problemen m.b.t. het gebruik van databases. Zorg er ook voor dat de webapplicatie minimale rechten heeft op de database en gevoelige informatie versleuteld opslaat. De impact van eventuele kwetsbaarheden is dan minimaal.

Controles:
- bekijk op welke manier de webapplicatie SQL-queries samenstelt
- controleer het effect van potentieel gevaarlijke karakters in de invoer zoals een puntkomma (';') en een apostrof
- controleer de rechten van het database-account waarvan de webapplicatie gebruik maakt

Wilhelmina van Pruisenweg 104
2595 AN Den Haag

Postbus 117
2501 CC Den Haag

T 070 888 7 555
E info@govcert.nl
I www.govcert.nl

GOVCERT.NL is het Cyber Security & Incident Response Team van de Nederlandse overheid. Vragen over de inhoud van deze factsheet kunt u richten aan info@govcert.nl of telefonisch via 070-8887555.