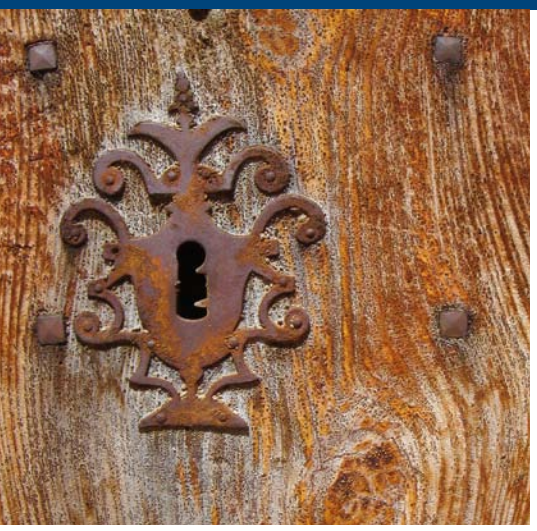


Waarom vraagt LinkedIn je Google-wachtwoord?

P. Ceelen MSc
is als junior adviseur werkzaam bij KPMG IT Advisory. Hij houdt zich onder andere bezig met opdrachten op het gebied van Identity & Access Management en de beveiliging van internettechnologie.

ceelen.pieter@kpmg.nl

Pieter Ceelen MSc

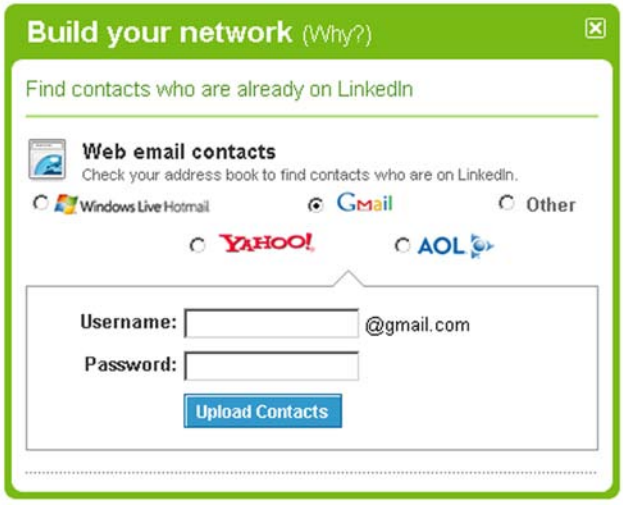
Met de opkomst van softwarediensten over het internet (SaaS) spelen er diverse nieuwe uitdagingen op het gebied van informatiebeveiliging. In [Chuno8] werd al beschreven dat SaaS-architecturen nieuwe beveiligingsrisico's creëren door de complexiteit van diensten en integratie van softwarediensten.
Indien men SaaS- en Web 2.0-concepten gaat toepassen, worden de beperkingen van de huidige internetinfrastructuur op het gebied van Identity & Access Management zichtbaar.
In dit artikel wordt een aantal opties voor de integratie van webdiensten bekeken, en worden de eraan gerelateerde beveiligingsrisico's besproken. Zowel huidige technieken als toekomstige ontwikkelingen op dit gebied passeren de revue.

Het delen van informatie tussen websites is slecht geregeld

Vanuit LinkedIn kun je contacten vanuit een webmailomgeving (bijvoorbeeld Gmail) importeren. Echter, om deze contactgegevens te delen tussen Gmail en LinkedIn moet je de Gmail-gebruikersnaam en het wachtwoord in een LinkedIn-pagina invoeren. Als je iets langer stilstaat bij dit fenomeen, komen er allerlei vragen naar boven: Waarom is dit nodig? Wat gebeurt er met dit wachtwoord? Leest LinkedIn ook mijn e-mail en Gmail calendar? Is het invoeren van dit wachtwoord veilig? Kan dit niet anders?

Dit voorbeeld laat zien dat er de noodzaak is om informatie te delen tussen verschillende webapplicaties. Applicatieontwikkelaars en site-eigenaren willen zoveel mogelijk informatie verzamelen zodat de applicaties zich zo intelligent mogelijk gedragen. Ook voor gebruikers zijn er voordelen, data-uitwisseling tussen websites maakt het mogelijk om informatie op één plek te beheren.

Indien in LinkedIn je Gmail-account gekoppeld wordt, dan krijg je de mogelijkheid om je contacten op één plek te beheren. De gekozen oplossing heeft één groot nadeel: gebruikers worden geacht inloggegevens van bijvoorbeeld Gmail niet af te staan aan andere websites. Indien gebruikers gewend raken aan het afgeven van de inloggegevens, dan kunnen criminelen dit gedrag misbruiken en wachtwoorden achterhalen door nep-websites op te zetten.



Figuur 1. **Screenshot van LinkedIn.**

Vanuit het feit dat twee grote spelers op het internet (LinkedIn en Google) deze oplossing gekozen hebben, kunnen we concluderen dat de huidige beveiligingsmechanismen ontoereikend zijn voor het delen van informatie. Met ontwikkelingen zoals SaaS en Web 2.0 zal de noodzaak informatie te delen tussen internetdiensten alleen maar toenemen. Deze zogenaamde 'mashups' maken het mogelijk om informatie en diensten naar eigen inzicht te combineren. Ook in zakelijke toepassingen zal dit probleem zich steeds vaker gaan voordoen. In de toekomst moet een SaaS-oplossing voor CRM (Customer Relationship Management) integreren en communiceren met je e-mail en je kalender.

Uit het LinkedIn-voorbeeld komt ook naar voren dat gebruikers meer controle willen hebben over de data die zij delen tussen verschillende webdiensten. Als gebruiker wil ik alleen mijn contacten delen en niet mijn e-mail en kalender.

Wat is SaaS en Web 2.0?
Web 2.0 is de trend dat websites interactiever worden en steeds meer worden gemaakt van door gebruikers aangeleverde content. Door de toepassing van Ajax (asynchronous JavaScript en XML) hoeft een webpagina niet ververst te worden om nieuwe data in te laden. Typische voorbeelden zijn social-networking websites en bijvoorbeeld de interface van Gmail.
SaaS staat voor Software as a Service. Dit is een softwaredistributiemodel waarbij de software als een onlinedienst wordt geleverd. Gebruik van de software vereist alleen een internetverbinding, de SaaS-provider zorgt voor onderhoud en beheer van de dienst.

In dit artikel zal geanalyseerd worden welke risico's en problemen spelen bij het toestaan van communicatie tussen websites. Daarnaast worden de huidige technische mogelijkheden en toekomstige ontwikkelingen behandeld.

Communicatie tussen websites levert beveiligingsrisico's op

Eén van de meest fundamentele beveiligingsmaatregelen op het internet is de scheiding tussen websites van verschillende domeinen. Een domein is een deel van een internetadres, bij het adres 'http://bank.nl/internetbankieren' hoort het domein 'bank.nl'.

Beveiligingsmaatregelen in je browser zorgen ervoor dat elke website JavaScript-code mag uitvoeren, maar deze scripts zijn zodanig beperkt dat ze geen data mogen benaderen van een ander domein. Ter illustratie, als gebruiker wil je niet dat een script van een website met het domein 'kpmg.nl' toegang heeft tot data van je internetbankierenomgeving op het domein 'bank.nl'.

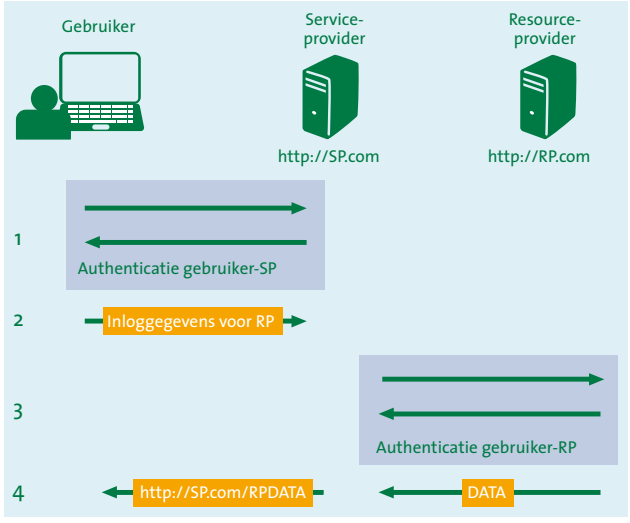
De scheiding tussen deze domeinen is in webbrowsers geïmplementeerd door middel van de zogenaamde Same-Origin Policy. De Same-Origin Policy zorgt ervoor dat data van een website beschermd worden. Data kunnen alleen worden uitgewisseld tussen websites indien zij dezelfde domeinnaam hebben en aan een aantal aanvullende technische voorwaarden voldoen. De Same-Origin Policy regelt dat een script op http://www.LinkedIn.com geen data kan benaderen van http://Gmail.com en dat een script van http://Gmail.com/contacts wel verbinding mag maken met http://Gmail.com/mail.

Ontwikkelaars van webapplicaties kunnen gebruikmaken van de Same-Origin Policy door data te plaatsen op verschillende domeinen. Een voorbeeld hiervan is het gebruik van http://maps.google.com en http://mail.google.com; doordat Google gekozen heeft voor gescheiden domeinen zal de browser geen communicatie toestaan tussen deze applicaties. Indien er gekozen was voor http://www.google.com/maps en http://www.google.com/mail, dan zou dit wel mogen. Het risico van de laatstgenoemde optie is dat een zwakheid (bijvoorbeeld een cross-site scripting aanval) in de maps-applicatie gevolgen heeft voor vertrouwelijkheid en integriteit van je mail.

Met de opkomst van mashups werd duidelijk dat het noodzakelijk is om deze scheiding tussen domeinnamen te omzeilen ([Bholo7]). Ontwikkelaars hebben een aantal oplossingen bedacht, die hierna besproken worden.

Proxy

In het voorbeeld van LinkedIn en Gmail wordt er gebruikgemaakt van een zogenaamde Proxy-oplossing. Figuur 2 bevat een schematische beschrijving van de Proxy-oplossing. In dit voorbeeld kan Gmail worden beschouwd als een resourceprovider; Gmail levert contactdata. LinkedIn is in het genoemde scenario de serviceprovider, LinkedIn levert de uiteindelijke dienst aan de eindgebruiker. Onderdeel van de service die geleverd wordt is 'resource consumption', waarbij data van een resourceprovider worden ingebed in de service.



Figuur 2. **Proxy-oplossing.**

In figuur 2 zijn vier fasen te onderscheiden:

1. In fase 1 authenticeert de gebruiker zich met de dienst waarvan hij gebruik wil gaan maken. Hij gebruikt de inloggegevens van de serviceprovider (LinkedIn).
2. Tijdens fase 2 geeft de gebruiker zijn inloggegevens van de resourceprovider (Gmail) aan de serviceprovider.
3. De serviceprovider logt met de verkregen inloggegevens in bij de resourceprovider.
4. De serviceprovider vraagt data aan de resourceprovider en stuurt deze door aan de gebruiker via de server van de serviceprovider. De data staan dus op de server van de serviceprovider (aangeduid met http://SP.com/RPdata).

Doordat de data beschikbaar is op http://sp.com/RPDATA kunnen scripts van http://sp.com hiervan gebruikmaken.

Vanuit beveiligingsoogpunt is de Proxy-oplossing geen ideale situatie. Alhoewel gebruikers een bewuste keuze maken om wachtwoorden af te staan is het de vraag of zij zich bewust zijn van de risico's die spelen:

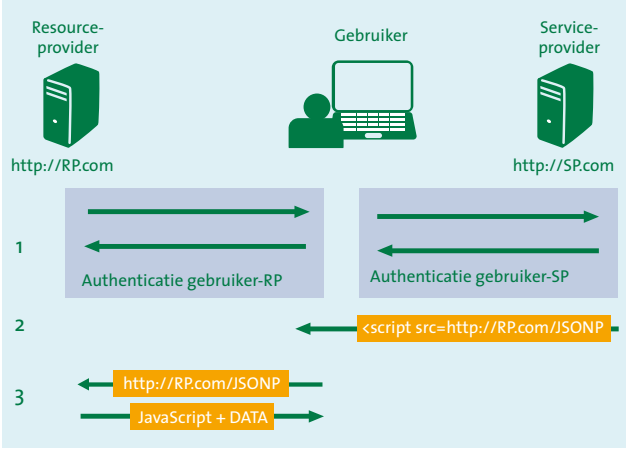
- Het afstaan van een wachtwoord geeft de serviceprovider volledige toegang tot het account bij de resourceprovider. De serviceprovider kan alle functionaliteit die een normale gebruiker heeft aanroepen, terwijl de gebruiker eigenlijk alleen autorisatie wil geven voor beperkte functionaliteit, zoals het toegankelijk maken van specifieke informatie.
- Een ander probleem ligt in het beheren van deze inloggegevens. Serviceproviders slaan de wachtwoorden op zodat zij altijd de actuele data uit kunnen lezen. Het spreekt voor zich dat het opslaan van wachtwoorden van een externe partij de nodige risico's met zich meebrengt.
- Een derde probleem met deze oplossing is het feit dat gebruikers normaal gesproken inloggegevens alleen moeten invoeren op de website die de inloggegevens verstrekt heeft. Wat gebeurt er als LinkedIn gaat vragen om de inloggegevens van internetbankieren? Gaan gebruikers deze afstaan omdat ze dit normaal achten?

Ook voor ontwikkelaars is deze situatie niet ideaal, er moet extra software ontwikkeld worden voor de Proxy-functionaliteit en voor het authenticeren bij resourceproviders.

JSONP

Als alternatief voor de bovengenoemde Proxy-oplossing is JSONP ([Remo05]) ontwikkeld. Binnen deze oplossing wordt gebruikgemaakt van de eigenschap dat een JavaScript-bestand ook op een andere domeinnaam mag staan. Figuur 3 toont schematisch aan hoe JSONP werkt.

1. In fase 1 authenticeert de gebruiker zich zowel bij de resourceprovider als bij de serviceprovider. Het betreft het reguliere inlogproces op beide websites.
2. De serviceprovider retourneert een HTML-pagina met daarin de instructie om een JavaScript-bestand van de resourceprovider op te halen.
3. De browser van de gebruiker haalt de data op van de resourceprovider; deze data zijn op een speciale manier 'ingepakt' in JavaScript. Door middel van samenwerking van scripts van de serviceprovider en de resourceprovider worden de data in de pagina verwerkt.



Figuur 3. **JSONP.**

Op het eerste gezicht lijkt deze oplossing erg elegant, gebruikersnamen en wachtwoorden hoeven niet te worden uitgewisseld. Het probleem met deze oplossing is dat de website van de serviceprovider het toestaat dat een JavaScript-bestand van een externe website (RP.com) wordt geladen. Dit bestand kan mogelijk kwaadaardige code bevatten.

De resourceprovider is in staat code uit te laten voeren op het domein van de serviceprovider. Indien de resourceprovider wil, kan hij bijvoorbeeld toegang krijgen tot gegevens opgeslagen bij de serviceprovider. Een mogelijke hack bij een resourceprovider kan dus direct invloed hebben op de veiligheid van serviceproviders.

Een ander probleem is dat een andere website, bijvoorbeeld evil.com, op eenzelfde manier ook deze data kan inlezen. JSONP is daarom niet geschikt voor privacygevoelige informatie.

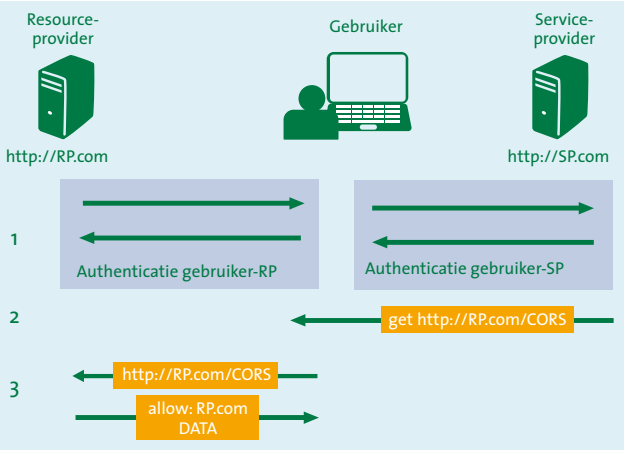
Huidige oplossingen zijn ontoereikend

Zoals hierboven aangegeven is het technisch mogelijk om data uit te wisselen tussen websites. De huidige oplossingen creëren echter nieuwe beveiligingsrisico's. Binnen de huidige mogelijkheden bestaat het risico dat een hacker die toegang heeft tot één website ook toegang kan krijgen tot alle verbonden websites. Zolang dit olievlekeffect niet voorkomen kan worden, zal men terughoudend zijn in het gebruik van deze oplossingen.

De toekomst: CORS

Met de ontwikkeling van een nieuwe webstandaard genaamd Cross-Origin Resource Sharing (CORS [Cros09]) worden de nadelen van de eerdergenoemde oplossingen aangepakt. CORS definieert mogelijkheden om uitzonderingen te maken op de Same-Origin Policy. Er is een mechanisme gekozen dat lijkt op JSONP, maar zonder de nadelen.

Figuur 4 bevat een schematische weergave van deze techniek.
1. In fase 1 authenticeert de gebruiker zich zowel bij de resourceprovider als bij de serviceprovider. Het betreft het reguliere inlogproces.
2. De serviceprovider retourneert een HTML-pagina met daarin de instructie om een JavaScript-bestand van de resourceprovider op te halen. Er wordt hierbij geen gebruik gemaakt van 'inpakken' zoals dat bij JSONP het geval is.
3. De resourceprovider ontvangt het request en levert de data aan. Naast de data wordt er een lijst met toegestane domeinnamen meegestuurd. De browser van de gebruiker zorgt ervoor dat alleen scripts van de domeinnamen uit het lijstje toegang krijgen tot deze data.



Figuur 4. **CORS.**

Door gebruik te maken van deze nieuwe functionaliteit is het 'inpakken' door middel van JavaScript zoals in JSONP niet meer noodzakelijk. Het grote voordeel hiervan is dat een resourceprovider geen code meer kan uitvoeren op het domein van de serviceprovider.

Door de toevoeging van de lijst met toegestane domeinnamen in het antwoord van de resourceprovider wordt het tweede nadeel van JSONP opgelost zonder afbreuk te doen aan de voordelen van dit mechanisme.

Er zijn al enkele browsers verkrijgbaar die deze manier van communicatie toestaan, onder andere Internet Explorer 8 en Firefox 3.5. De verwachting is dat applicaties de mogelijkheid gaan bieden om zelf de lijsten met toegestane domeinnamen te beheren. In theorie bied CORS de mogelijkheden om als gebruiker zelf aan te geven dat je alleen je contacten wil delen met LinkedIn en dat e-mail en kalenderdata alleen gedeeld mogen worden met bijvoorbeeld een CRM-pakket.

Conclusie

Proxy
+ Geen risico's voor serviceproviderdomein
– Geen controle over welke functionaliteit wordt ontsloten
– Serviceprovider moet accountgegevens beheren en beveiligen
– Ontwikkeling en beheer van Proxy-functionaliteit en authenticatie bij resourceproviders

JSONP
+ Transparante oplossing voor eindgebruikers
+ Weinig aanpassingen in bestaande websites
– Serviceprovider loopt risico's indien een resourceprovider gecompromitteerd is
– Resourceprovider heeft geen invloed op welke websites de informatie gebruiken

CORS
+ Lost bovengenoemde nadelen en risico's op
– Nog geen wijdverbreide browserondersteuning

Zoals in [Chuno8] aangegeven vereist de overstap naar SaaS-oplossingen een gedegen analyse van risico's voor de gehele dataketen. In dit artikel is ingezoomd op de technische risico's die spelen bij de integratie van webdiensten. Het blijkt dat op dit gebied de huidige oplossingen beveiligingrisico's introduceren die niet eenvoudig gemitigeerd kunnen worden.

De introductie van CORS biedt mogelijkheden om op een gecontroleerde manier data te delen tussen verschillende webdiensten. Het is een technologische verbetering op het gebied van access management van webdiensten. Aangezien ondersteuning voor CORS al aanwezig is in enkele browsers, is het wachten op de eerste toepassing van dit concept op grote schaal.

Literatuur

[Bholo7] S. Bhola, S. Chari en M. Steiner, *Security for Web 2.0 application scenarios: Exposures, Issues and Challenges*, Workshop Web 2.0 Security & Privacy, 2007.
[Chuno8] Mike Chung, *Informatiebeveiliging versus SaaS*, Compact 2008/4.
[Croso9] *Cross-Origin Resource Sharing – W3C Working Draft 17 March 2009*, http://www.w3.org/TR/access-control/.
[Remoo5] *Remote JSON – JSONP*, http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/.