6

COMMUNICATIE

De gave om duidelijk te schrijven zou wel eens de belangrijkste vaardigheid kunnen zijn voor iemand in een open source-omgeving. Op de lange duur speelt dit een grotere rol dan programmeertalent. Een uitstekende programmeur met waardeloze communicatieve vaardigheden kan maar één ding tegelijk. En zelfs daarvoor weet hij wellicht niet voldoende aandacht te trekken. Maar een waardeloze programmeur met uitstekende communicatieve vaardigheden kan vele anderen coördineren en overhalen verschillende dingen te doen en daarmee een aanzienlijke bijdrage leveren aan de richting en de voortgang van het project.

Er lijkt weinig verband te bestaan tussen de vaardigheid om goede code te schrijven en de vaardigheid om met je medemensen te communiceren. Er is wel een enig verband tussen het goed kunnen programmeren en technische kwesties goed kunnen omschrijven, maar het beschrijven van technische kwesties maakt slechts een zeer klein onderdeel uit van de communicatie binnen een project. Veel belangrijker is de vaardigheid je in te leven in de doelgroep, je eigen posts en commentaren te zien zoals anderen ze zien en ervoor te zorgen dat anderen hun eigen posts met dezelfde objectiviteit zien. Net zo belangrijk is kunnen herkennen wanneer een bepaald medium of communicatiemiddel niet langer goed werkt, misschien omdat het niet meer past bij de omvang van het project wanneer het aantal gebruikers groter wordt, en in dat geval de tijd nemen hier iets aan te doen.

Al deze punten liggen in theorie voor de hand, maar in de praktijk bemoeilijkt de verwarrende diversiteit van omgevingen waarin open source-software wordt ontwikkeld, wat betreft zowel doelgroepen als communicatiemechanismen, de situatie. Moet een bepaald idee in een post op de mailinglijst worden gezet, als een melding voor de bug tracker of als een opmerking in de code? En hoeveel kennis mag u, bij het beantwoorden van vragen op een publiek forum, aannemen dat de lezer heeft, uitgaande van het feit dat de ‘lezer’ niet alleen degene is die de vraag stelt maar ook iedereen die uw reactie leest? Hoe kan er constructief contact zijn tussen ontwikkelaars en gebruikers, zonder te worden overspoeld door verzoeken om functies, foutieve bugrapporten en gekwebbel? Hoe kunt u zien wanneer een medium zijn maximale capaciteit heeft bereikt en wat moet u daaraan doen?

Oplossingen voor deze problemen lossen vaak maar een deel ervan op, omdat iedere oplossing uiteindelijk achterhaald is als de omvang van het project toeneemt of de structuur verandert. Ze worden ook vaak ad hoc toegepast, omdat het geïmproviseerde reacties zijn op dynamische situaties. Alle deelnemers moeten zich bewust zijn van wanneer en hoe de communicatie vast kan lopen en een bijdrage leveren aan de oplossing. Mensen hierbij helpen is een belangrijk onderdeel van het managen van een open source-project. In de onderstaande gedeelten wordt besproken hoe u met uw eigen communicatie moet omgaan en hoe u ervoor kunt zorgen dat het onderhoud van de communicatiemechanismen voor iedereen binnen het project prioriteit heeft.[22]

6.1 U BENT WAT U SCHRIJFT

Denk hier eens over na: het enige dat mensen over u weten, weten ze door wat u schrijft of wat anderen over u schrijven. Misschien bent u een briljant, scherpzinnig en charismatisch persoon, maar als uw e-mails onsamenhangend en ongestructureerd zijn, zullen mensen denken dat u dat ook bent. Of misschien bent u juist wel een onsamenhangend en ongestructureerd persoon, maar dat hoeft niemand ooit te weten te komen als uw posts helder en informatief zijn.

Extra aandacht besteden aan uw schrijfstijl loont uiteindelijk zeer de moeite. Dit verhaal is afkomstig van Jim Blandy, een ervaren programmeur van open source-software.

In 1993 werkte ik voor de Free Software Foundation. We waren bezig met het testen van bètaversie 19 van GNU Emacs. We maakten ongeveer iedere week een bètarelease. Men probeerde die dan uit en stuurde ons bugrapporten toe. Er was één man die niemand van ons persoonlijk had ontmoet maar die geweldig werk verzette. Zijn bugrapporten waren altijd duidelijk en leidden ons recht op het probleem af. En als hij zelf met een fix kwam, klopte die meestal precies. Hij was echt super.

Voordat de FSF echter code kan gebruiken die door iemand anders is geschreven, moet er wat juridisch papierwerk worden gedaan waarmee mensen de auteursrechten van hun code overdragen aan de FSF. Code aannemen van een totale onbekende en zomaar gebruiken, zorgt gegarandeerd voor een juridische ramp.
Ik e-mailde onze vriend daarom de formulieren met de boodschap "Hier zijn wat papieren die we nodig hebben. U ondertekent dít formulier, vraagt uw werkgever dát formulier te ondertekenen en we kunnen uw fixes gaan gebruiken. Bij voorbaat dank."

Hij antwoordde: "Ik heb geen werkgever."

Dus zei ik "Ok, prima, laat uw universiteit het dan ondertekenen en stuur het terug."

Na een tijdje schreef hij terug en zei, "Nou ja, eh ... ... ik ben pas dertien en ik woon nog bij mijn ouders."

Omdat de jongen niet schreef als een dertienjarige dacht iedereen dat hij veel ouder was. Hieronder volgens een paar aanwijzingen voor uw schrijfstijl waarmee ook u een goede indruk kan maken.

Structuur en opmaak
Laat u niet verleiden alles op te schrijven als een sms-bericht. Schrijf in complete zinnen, met een hoofdletter aan het begin van iedere zin, en begin waar nodig een nieuwe paragraaf. Dit is in e-mails en andere schrijfsels zeer belangrijk. Bij IRC's of soortgelijke kortstondige forums is het meestal geen probleem om hoofdletters weg te laten, verkorte vormen van veelgebruikte uitdrukkingen te gebruiken enz. Neem deze gewoonte alleen niet over in formelere en langer durende forums. E-mails, documentatie, bugrapporten en andere documenten die bedoeld zijn voor permanent gebruik moeten een samenhangende en beschrijvende structuur hebben. Dat is niet omdat het per se goed is om willekeurige regels te volgen, maar omdat deze regels *niet* willekeurig zijn Ze zijn geëvolueerd tot hun huidige vorm omdat ze tekst leesbaarder maken. En dat is de reden dat u ze moet toepassen. Leesbaarheid is niet alleen belangrijk omdat het betekent dat mensen begrijpen wat u schrijft, maar ook omdat het u neerzet als iemand die de tijd neemt om duidelijk te communiceren; dat wil zeggen, iemand die de moeite waard is om naar te luisteren.

Met name ten aanzien van e-mails zijn er voor ontwikkelaars van open source bepaalde ongeschreven regels:

Stuur alleen e-mails met platte tekst, geen HTML, RichText of andere bestandsindelingen die niet leesbaar zijn voor e-mailprogramma's die alleen platte tekst kunnen lezen. Stel uw regels zo in dat ze ongeveer 72 karakters lang zijn. Kom niet boven de 80 karakters, dit is de *de facto* standaardbeeldschermbreedte (dat wil zeggen dat mensen wel bredere beeldschermen kunnen hebben maar geen smallere). Door uw tekst zo op te maken dat het iets *minder* is dan 80 karakters breed houdt u plek over voor een aantal extra aanhalingstekens die in antwoorden van anderen kunnen worden ingevoegd zonder dat uw tekst moet worden opgemaakt.

Gebruik echte regelterugloop. Sommige e-mailprogramma's gebruiken een soort onechte regelterugloop, waarbij tijdens het schrijven van een e-mail uw scherm regelafbrekingen laat zien die er in werkelijkheid niet zijn. Wanneer de e-mail wordt verzonden bevat deze niet de regelafbrekingen die u dacht dat hij had en zal de tekst op sommige beeldschermen anders teruglopen. Wanneer uw e-mailprogramma onechte regelafbreking gebruikt, zoek dan naar een instelling om de harde afbrekingen tijdens het schrijven zichtbaar te maken.

Als u schermoutput, stukjes code of andere opgemaakte tekst opneemt, laat dit dan duidelijk in de opmaak zien, zodat zelfs de luie lezer de grens tussen uw eigen schrijfsels en het aangehaalde materiaal makkelijk kan herkennen. Toen ik met dit boek begon, had ik nooit gedacht dit te moeten adviseren, maar ik heb recentelijk op een aantal mailinglijsten gezien dat mensen de tekst van verschillende bronnen door elkaar gooiden zonder duidelijk te maken welke tekst waarbij hoort. Het effect hiervan is erg frustrerend. Hun posts zijn hierdoor aanzienlijk moeilijker te begrijpen en die mensen komen daardoor eerlijk gezegd een beetje ongeorganiseerd over.

Wanneer u een e-mail van iemand anders citeert, voeg dan uw antwoord toe op de plek waar het bij hoort, zo nodig op verschillende plaatsen, en verwijder de gedeeltes van de e-mail die u niet gebruikt. Wanneer u een korte reactie schrijft die betrekking heeft op iemands gehele e-mail kunt u een *top-post* maken (d.w.z. dat u uw antwoord boven de aangehaalde tekst van de e-mail zet). In andere gevallen zou u het betreffende deel van de oorspronkelijke tekst eerst moeten aanhalen, gevolgd door uw reactie.

Denk goed na over de onderwerpregels van nieuwe e-mails. Het is de belangrijkste regel van uw e-mail, omdat het de anderen binnen het project gelegenheid geeft te beslissen of ze wel of niet verder willen lezen. Moderne e-mailprogramma's kunnen groepen bij elkaar behorende berichten in threads zetten, waarbij e-mails niet alleen aan de hand van een algemeen onderwerp kunnen worden gegroepeerd, maar ook aan de hand van andere koppen (die soms niet worden weergegeven). Logisch gevolg hiervan is dat u, wanneer een thread afdwaalt naar een nieuw onderwerp, het onderwerpveld zou kunnen (en moeten) aanpassen aan het nieuwe onderwerp. De integriteit van de thread blijft bewaard, als gevolg van de vorige koppen, maar het nieuwe onderwerp laat mensen die een overzicht van de thread bekijken, weten dat het onderwerp is veranderd. Dat is ook de reden dat u, wanneer u een nieuw onderwerp wilt beginnen, een nieuwe e-mail moet maken en niet moet antwoorden op een bestaande e-mail en het onderwerp veranderen. Uw e-mail zou anders nog steeds worden ondergebracht in dezelfde thread als waarop u antwoordt, waardoor mensen ten onrechte kunnen denken dat hij over hetzelfde onderwerp gaat. Ook hier is het nadeel niet alleen verspilling van hun tijd, maar ook de kleine beschadiging van uw reputatie als iemand die goed is in het gebruik van communicatiemiddelen.

Inhoud

Goed opgemaakte e-mails trekken lezers aan, maar de inhoud houdt ze vast. Natuurlijk zijn er geen vaste regels die goede inhoud garanderen, maar er zijn wel een paar uitgangspunten die ertoe bijdragen.

Maak het uw lezers gemakkelijk. Er gaat ontzettend veel informatie om in een actief open source-project en van de lezers kan niet worden verwacht dat ze overal van op de hoogte zijn. Sterker nog, er kan zelfs niet van hen worden verwacht dat ze weten hoe ze ervan op de hoogte moeten blijven. Waar mogelijk moet uw post informatie bevatten in een vorm die het meest geschikt is voor de lezers. Als u twee extra minuten moet spenderen aan het opsporen van de URL van een bepaalde thread in het archief van de mailinglijst om uw lezers deze moeite te besparen, dan is het de moeite waard. Wanneer u vijf of tien minuten extra kwijt bent aan het samenvatten van de conclusies tot dan toe van een complexe thread, om mensen daarmee een context te geven voor uw e-mail, doe dat dan. U moet het zo zien: hoe succesvoller een project, des te gunstiger de verhouding tussen het aantal lezers ten opzichte van het aantal schrijvers op de forums. Wanneer iedere post wordt gezien door *n* personen, dan gaat het nut van het besteden van extra moeite aan het besparen van tijd voor deze mensen ook omhoog wanneer *n* omhoog gaat. Ook zullen mensen die zien dat u zichzelf deze taak oplegt zelf ook hun best hiervoor doen in hun eigen berichten. In het ideale geval is het resultaat een stijging van de algehele efficiëntie van het project: wanneer er gekozen kan worden tussen *n* mensen die moeite moeten doen, of één persoon, dan is de laatste het voordeligst voor het project.
Verval niet in overdrijving. Overdrijven in online posts is een klassieke bewapeningswedloop. Een persoon meldt bijvoorbeeld een bug en hij is bang dat de ontwikkelaars er te weinig aandacht aan zullen besteden. Daarom omschrijft hij de bug als een ernstig probleem dat het hele project schaadt en waardoor hij, en al zijn vrienden/collega's/familieleden, de software niet productief kunnen gebruiken, terwijl het in feite slechts om een kleine ergerlijkheid gaat. Overdrijving wordt echter niet alleen door gebruikers gebezigd, ook programmeurs hebben er een handje van tijdens technische discussies, met name wanneer de onenigheid betrekking heeft op smaak en niet op correcte code:

"Als we dat doen wordt de code compleet onleesbaar. Het onderhoud ervan wordt een nachtmerrie, in tegenstelling tot het voorstel van J. Random ..."

Hetzelfde gevoel kan zelfs *sterker* worden overgebracht met minder scherp taalgebruik:

"Het werkt wel, maar volgens mij is het minder geschikt wat betreft leesbaarheid en onderhoudsgemak. Het voorstel van J. Random voorkomt dit probleem omdat het ..."

U zult niet helemaal zonder overdrijving kunnen en over het algemeen hoeft u dat ook niet te proberen. Vergeleken met andere vormen van miscommunicatie is overdrijving niet schadelijk voor het hele project, alleen voor de overdrijver. De ontvangers weten er echt wel mee om te gaan, alleen de zender verliest iedere keer weer een beetje van zijn geloofwaardigheid. Daarom zou u, ten behoeve van uw eigen invloed binnen het project, moeten proberen gematigd te blijven in uw uitspraken. Ook nemen mensen u dan serieus wanneer u *wel* een belangrijk punt naar voren brengt.

Kijk uw bericht twee keer na. Lees ieder bericht dat langer is dan een gemiddelde paragraaf nog eens van A tot Z door voordat u het verstuurt, en dan nog een keer. Dit is een bekend advies voor mensen die een schrijfcursus hebben gevolgd, maar voor online discussies is het extra belangrijk. Omdat in veel gevallen het schrijfproces van online berichten dikwijls wordt onderbroken (tijdens het schrijven moet u misschien andere e-mails nalezen, webpagina's bezoeken, een commando runnen om de output te genereren enz.) kunt u uw verhaallijn makkelijk uit het oog verliezen. Berichten die met veel onderbrekingen zijn geschreven en niet meer gecontroleerd zijn, zijn dikwijls als zodanig te herkennen, vaak tot grote ergernis van hun auteurs (dat hopen we tenminste). Neem de tijd om na te kijken wat u verstuurt. Hoe beter de structuur van uw posts is, des te meer ze zullen worden gelezen.

Toon

Na het schrijven van duizenden berichten zult u merken dat uw schrijfstijl steeds beknopter wordt. Dit lijkt normaal te zijn voor de meeste technische forums en daar is in principe niets mis mee. Een bepaalde mate van beknoptheid, die in normale sociale omgang niet acceptabel zou zijn, is nou eenmaal de norm voor hackers van

open source-software. Hieronder een reactie die ik ooit kreeg op een mailinglijst over open source-software voor contentmanagement, volledig geciteerd:

```
Kun je misschien wat meer vertellen over de problemen waar je
precies tegenaan liep, etc.?

Ook:

Welke versie van Slash gebruik je? Dit kon ik niet opmaken uit
je oorspronkelijke bericht.

Hoe heb je de source van apache/mod_perl precies gebouwd?

Heb je de Apache 2.0-patch die gepost is op slashcode.com al ge-
probeerd?

Shane
```

Dat is nog eens beknopt! Geen begroeting, behalve zijn naam geen afsluiting en het bericht bestaat uit niet meer dan een reeks vragen die zo compact mogelijk zijn gesteld. En de enige verklarende zin was ook nog kritiek op mijn oorspronkelijke bericht. Maar toch was ik blij met Shane's e-mail en het feit dat hij beknopt was, zag ik alleen als teken dat hij het druk had. Alleen al het feit dat hij vragen stelde, in plaats van mijn post te negeren, betekende dat hij bereid was tijd te besteden aan mijn probleem.

Maar zullen alle lezers positief reageren op deze manier van schrijven? Niet per definitie, dat hangt af van de persoon en de context. Wanneer iemand bijvoorbeeld net een fout heeft moeten toegeven - misschien heeft hij een bug geschreven - en u weet uit het verleden dat deze persoon soms onzeker is, dan kunt u nog steeds een korte reactie schrijven, maar zorg er ook voor dat u laat merken begrip te hebben voor zijn gevoelens. Het grootste deel van uw reacties kan een korte en technische analyse van de situatie zijn, zo kort en zakelijk als u maar wilt. Maar sluit uw bericht altijd af met iets waarmee u aangeeft dat u misschien wel kortaf maar niet onvriendelijk bent. Wanneer u bijvoorbeeld een hele waslijst met adviezen heeft gegeven over hoe iemand een bug moet herstellen, sluit dan af met "Succes, <uw naam>" om aan te geven dat u hem het beste toewenst en niet boos bent. Een strategisch geplaatste smiley of andere emoticon kan vaak ook genoeg zijn om uw gesprekspartner gerust te stellen.

Het lijkt misschien vreemd om evenveel aandacht te besteden aan iemands gevoelens als aan wat ze inhoudelijk zeggen, maar gevoelens zijn nou eenmaal van invloed op de productiviteit. Gevoelens zijn ook om andere redenen belangrijk, maar zelfs als we ons beperken tot puur pragmatische redenen zullen we merken dat ongelukkige mensen slechtere software schrijven, en ook nog eens minder. Door de beperkingen van de meeste elektronische media hebben we meestal geen duidelijk beeld van hoe een persoon zich voelt. U zult dat zo goed mogelijk moeten inschatten op basis van a) hoe de meeste mensen zich in een dergelijke situatie zouden voelen en b) wat u over deze persoon weet uit voorgaande gesprekken. Sommige mensen hebben de voorkeur voor een meer afstandelijke benadering en gaan met iedereen om op basis van de indruk die ze maken. Het idee daarachter is dat als een deelnemer zijn gevoelens niet direct aangeeft iemand anders niet het recht heeft hem wel op die manier te behandelen. Ik ben het om een aantal redenen niet met deze benadering eens. Eén ervan is dat mensen zich in het dagelijks leven ook niet zo gedragen. Dus waarom zouden ze dat online wel doen? Een tweede reden is dat mensen op publieke forums - waar de meeste gesprekken plaatsvinden - nog minder geneigd zijn hun gevoelens te tonen dan in hun privéleven. Of beter gezegd, mensen zijn vaak wel bereid hun gevoelens ten opzichte van anderen te uiten, zoals dankbaarheid of verontwaardiging, maar niet hun innerlijke gevoelens, zoals onzekerheid of trots. Toch werken de meeste mensen beter wanneer ze weten dat anderen zich bewust zijn van hun gemoedstoestand. Door aandacht te besteden aan kleine aanwijzingen kunt u meestal wel een juiste inschatting maken. Daardoor kunt u mensen beter motiveren om betrokken te blijven dan anders het geval zou zijn.

Ik bedoel natuurlijk niet dat u altijd de rol van therapeut van de hele groep op zich moet nemen door iedereen steeds te helpen met zichzelf in het reine te komen. Maar door voldoende aandacht te besteden aan patronen in het gedrag van mensen kunt u inzicht krijgen in hun persoonlijkheid, zelfs wanneer u ze nooit persoonlijk ontmoet. En door zich bewust te zijn van de toon van uw schrijfsels kunt u verrassend veel invloed hebben op hoe anderen zich voelen, wat uiteindelijk alleen maar goed kan zijn voor het project.

Onbeleefd gedrag herkennen

Een van de belangrijkste kenmerken van de open source-cultuur is duidelijke ideeën over wat wel en niet wordt gezien als onbeleefd taalgebruik. Hoewel de onderstaande discussies niet uniek zijn voor de ontwikkeling van open source-software en zelfs niet van software in het algemeen - ze zullen iedereen die werkt in de vakgebieden wiskunde, de exacte wetenschappen of techniek zelfs bekend voorkomen - is open source-software, met zijn soms onduidelijke grenzen en constante toestroom van nieuwkomers, een omgeving waar mensen vaak tegen normen aanlopen waarmee ze bekend zijn.

Laten we beginnen met de dingen die *niet* onbeleefd zijn:

Technische kritiek, zelfs als dit direct en met weinig tact wordt gebracht, is niet onbeleefd. Het kan zelfs als een compliment worden beschouwd: de kritiek impliceert dat het lijdend voorwerp ervan serieus moet worden genomen en de moeite waard is tijd aan te besteden. Dat wil zeggen dat hoe groter de kans zou zijn dat iemands post gewoon wordt genegeerd, des te groter het compliment is wanneer iemand de tijd neemt kritiek te leveren, tenzij de kritiek natuurlijk vervalt in een *persoonlijke* aanval of een andere vorm van overduidelijke onbeleefdheid.

Botte vragen zonder enige omlijsting, zoals Shane's vraag in de hiervoor geciteerde e-mail, zijn evenmin onbeleefd. Vragen die in een andere context misschien koel, gekunsteld of zelfs onecht kunnen overkomen, zijn vaak serieus bedoeld en hebben geen andere bedoeling dan zo snel mogelijk informatie te krijgen. De beroemde

vraag van een technische dienst "Zit de stekker van uw computer in het stopcontact?" is hier een klassiek voorbeeld van. De vraagsteller moet echt weten of de stekker er in zit en was het al na een paar dagen dit werk te hebben gedaan zat om zijn vraag steeds weer in te leiden met beleefdheden ("Neemt u mij niet kwalijk, ik wil u graag een paar eenvoudige vragen stellen om een aantal mogelijkheden uit te kunnen sluiten. Sommige daarvan lijken misschien wel erg simpel, maar ik vraag u geduld met me te hebben ..."). Op dit punt neemt hij niet meer de moeite zijn vraag mooi in te kleden en vraagt direct op de man af: "Zit de stekker in het stopcontact?" Vergelijkbare vragen worden ook op mailinglijsten van open source-softwareprojecten steeds weer gesteld. De bedoeling is niet om de ontvanger te beledigen, maar gewoon om snel de meest voor de hand liggende (en misschien wel meest voorkomende) verklaringen uit te kunnen sluiten. Ontvangers die dit begrijpen en overeenkomstig reageren, worden gewaardeerd om hun ruimdenkende instelling zonder hier verder woorden aan vuil te maken. Ontvangers die hier slecht op reageren, hoeven hier echter nou ook niet voor op hun kop te krijgen. Het is gewoon een botsing tussen twee culturen waar niemand schuld aan heeft. Leg op een vriendelijke manier uit dat u met uw vraag of kritiek geen bijbedoelingen had. Het was alleen bedoeld om op een zo efficiënt mogelijk manier informatie te krijgen of te geven.

Wat is dan wel onbeleefd?

Om dezelfde reden dat gedetailleerde technische kritiek een vorm van complimenteren kan zijn, kan het niet geven van kwalitatieve kritiek een vorm van belediging zijn. Ik bedoel hier niet het eenvoudigweg negeren van iemands werk, voorstel, codewijziging, gerapporteerde issue of wat dan ook. Tenzij u vooraf hebt beloofd om een gedetailleerde reactie te geven, is het over het algemeen geen probleem wanneer u helemaal niet reageert. Mensen zullen er vanuit gaan dat u geen tijd had om te reageren. Maar als u *wel* reageert, raffel het dan niet af. Neem de tijd om alles goed te analyseren, geef waar mogelijk concrete voorbeelden, ga in de archieven graven om gerelateerde posts uit het verleden te vinden enz. Als u hiervoor geen tijd hebt maar wel een korte reactie moet geven, wees dan open over deze tekortkoming ("Ik denk dat dit onderwerp al eerder aan bod is geweest, maar helaas heb ik geen tijd gehad om ernaar te zoeken, sorry"). Het belangrijkste dat onderkend moet worden, is het bestaan van een cultuur. Dat kan door erin mee te gaan of door openlijk toe te geven een keer de fout in te zijn gegaan. Wat er ook gebeurt, de norm wordt hierdoor versterkt. Maar als u niet aan de norm voldoet, terwijl u niet toegeeft *waarom* niet, komt het over alsof het onderwerp (en de mensen die eraan meedoen) niet de moeite waard was om tijd aan te besteden. U kunt beter later merken dat uw tijd kostbaar is door bondig te zijn dan lui.

Er zijn natuurlijk nog vele andere vormen van onbeleefd gedrag, maar de meeste daarvan zijn niet specifiek voor softwareontwikkeling en kunnen met wat gezond verstand worden voorkomen. Zie ook het gedeelte 'Grofheden in de kiem smoren' in Hoofdstuk 2*, Aan de gang*, wanneer u dat nog niet hebt gedaan.

Het gezicht

Er is een gedeelte van het menselijk brein dat speciaal is gereserveerd voor het herkennen van gezichten. Dit staat ook wel bekend als het 'fusiforme gezichtsherkenningsgebied' en de eigenschappen hiervan zijn voor het grootste deel aangeboren, niet aangeleerd. Het blijkt dat het herkennen van mensen afzonderlijk zo essentieel is om te overleven, dat we daar in de loop van de evolutie speciale 'hardware' voor hebben ontwikkeld.

Samenwerking via internet is daarom psychologisch gezien ongewoon, omdat er nauw wordt samengewerkt tussen mensen die elkaar bijna nooit leren kennen via de meest natuurlijke en intuïtieve methodes: gezichtsherkenning op de eerste plaats, maar ook stemgeluid, houding enz. Probeer, om dit te compenseren, om overal een consistente *schermnaam* te gebruiken. U kunt hiervoor het eerste deel van uw e-mailadres (het deel voor het @-karakter), uw IRC-gebruikersnaam, uw committernaam, uw gebruikersnaam als issue tracker enz. gebruiken. Deze naam is uw online 'gezicht': een korte herkenbare reeks karakters die wat betreft de bedoeling ervan overeenkomsten heeft met uw echte gezicht, hoewel deze helaas niet dezelfde ingebouwde hardware in onze hersenen stimuleert als een echt gezicht.

Uw schermnaam kan bijvoorbeeld een intuïtieve lettercombinatie zijn uit uw eigen naam. Die van mij is bijvoorbeeld 'kfogel'. In sommige situaties gaat deze sowieso vergezeld van uw volledige naam, bijvoorbeeld in e-mailvelden:

```
Van: "Karl Fogel" <kfogel@whateverdomain.com>
```

In feite zijn er twee dingen te zien in dit voorbeeld. Zoals eerder al gezegd komt de schermnaam op een intuïtieve manier overeen met de werkelijke naam. Maar de echte naam is ook *echt*. Dat wil zeggen dat het niet zomaar een bedachte benaming is zoals:

```
Van: "Wonder Hacker" <wonderhacker@whateverdomain.com>
```

Paul Steiner publiceerde op 5 juli 1993 een beroemd geworden cartoon in *The New Yorker*, waarin een hond te zien is die is ingelogd op een computerterminal en een andere hond samenzweerderig vertelt: "Op internet ziet niemand dat je een hond bent." Deze gedachtegang is waarschijnlijk de reden achter de vele zelfverheerlijkende, hip bedoelde online identiteiten die mensen zichzelf toekennen. Alsof een naam als 'Wonder Hacker' er werkelijk voor zorgt dat mensen geloven dat deze persoon ook een geweldige hacker *is*. Maar één feit blijft: ook als niemand weet dat u een hond bent, blijft u een hond. Een fantastische online identiteit heeft nog nooit indruk gemaakt op een lezer. Mensen denken in plaats daarvan dat het bij u meer om uw imago dan om de inhoud gaat, of dat u gewoon onzeker bent. Gebruik uw echte naam voor al uw communicatie. Als u om wat voor reden dan ook anoniem wilt blijven, bedenk dan een naam die heel normaal klinkt en gebruik deze consequent.

Naast een consistent online gezicht zijn er een paar dingen die het gezicht nog aantrekkelijker kunnen maken. Als u een officiële titel hebt, zoals 'doctor', 'professor' of 'directeur', loop er dan niet mee te koop. Vermeld hem niet eens, behalve wanneer het direct relevant is voor de discussie. In het hackerswereldje in het alge-

meen, en in het bijzonder binnen de vrijesoftwarecultuur, wordt het pronken met titels gezien als arrogantie en een teken van onzekerheid. Het is geen probleem als uw titel vermeld staat in de standaarde-mailhandtekening onderaan ieder bericht dat u verstuurt. Gebruik deze alleen nooit om uw gelijk te halen in een discussie; dit heeft gegarandeerd een averechtse uitwerking. U wilt dat mensen u respecteren als persoon, niet om uw titel.

En nu we het toch over e-mailhandtekeningen hebben: houd ze klein en smaakvol, of nog beter, gebruik ze helemaal niet. Voorkom eveneens lange juridische disclaimers aan het einde van iedere e-mail, met name wanneer daar opvattingen in staan die niet passen bij deelname aan een open source-softwareproject. De volgende klassieker uit het genre staat aan het eind van iedere post van een bepaalde gebruiker van een publieke mailinglijst waarvan ik ook lid ben:

```
IMPORTANT NOTICE

If you have received this e-mail in error or wish to read our
e-mail disclaimer statement and monitoring policy, please refer
to the statement below or contact the sender.

This communication is from Deloitte & Touche LLP.  Deloitte &
Touche LLP is a limited liability partnership registered in
England and Wales with registered number OC303675.  A list of
members' names is available for inspection at Stonecutter Court,
1 Stonecutter Street, London EC4A 4TR, United Kingdom, the firm's
principal place of business and registered office.  Deloitte &
Touche LLP is authorised and regulated by the Financial Services
Authority.

This communication and any attachments contain information
which is confidential and may also be privileged.  It is for the
exclusive use of the intended recipient(s).  If you are not the
intended recipient(s) please note that any form of disclosure,
distribution, copying or use of this communication or the
information in it or in any attachments is strictly prohibited
and may be unlawful.  If you have received this communication in
error, please return it with the title "received in error" to
IT.SECURITY.UK@deloitte.co.uk then delete the email and destroy
any copies of it.

E-mail communications cannot be guaranteed to be secure or
error free, as information could be intercepted, corrupted,
amended, lost, destroyed, arrive late or incomplete, or contain
viruses.  We do not accept liability for any such matters or
their consequences.  Anyone who communicates with us by e-mail
is taken to accept the risks in doing so.

When addressed to our clients, any opinions or advice contained
in this e-mail and any attachments are subject to the terms and
conditions expressed in the governing Deloitte & Touche LLP
client engagement letter.

Opinions, conclusions and other information in this e-mail and
any attachments which do not relate to the official business of
the firm are neither given nor endorsed by it.
```

Voor iemand die zich zo nu en dan laat zien om een vraag te stellen, lijkt deze gigantische disclaimer misschien wat onnozel, maar het kan waarschijnlijk niet echt kwaad. Als deze persoon echter actief wil participeren in het project kan zo'n blok tekst een sluipende impact hebben. Hij zendt minstens twee mogelijk destructieve signalen uit: ten eerste beheerst deze persoon zijn technische middelen niet volledig (hij zit vast in een bedrijfse-mailsysteem dat irritante boodschappen toevoegt aan het eind van iedere e-mail en hij weet niet hoe hij dat moet omzeilen) en ten tweede heeft hij weinig steun vanuit zijn organisatie voor zijn activiteiten op het gebied van open source-software. De organisatie heeft hem duidelijk niet rechtstreeks verboden om berichten te plaatsen op publieke mailinglijsten, maar zorgt er wel voor dat zijn posts er beslist onvriendelijk uitzien, alsof het risico op het bekendmaken van vertrouwelijke informatie alle andere prioriteiten overtreft.

Als u voor een organisatie werkt die erop staat dat zulke tekstblokken aan alle uitgaande berichten worden toegevoegd, overweeg dan om een gratis e-mailaccount speciaal voor het project te nemen, bijvoorbeeld bij gmail.google.com, www.hotmail.com of www.yahoo.com.

6.2 DE GEBRUIKELIJKE VALKUILEN OMZEILEN

Plaats geen nutteloze posts

Een veel voorkomende valkuil bij deelname aan projecten is denken dat u op alles moet reageren. Dat hoeft niet. Allereerst lopen er meestal meerdere threads tegelijk, die u niet allemaal kunt volgen, in ieder geval niet als het project de eerste fase van enkele maanden achter zich heeft. Ten tweede is er op veel dingen die mensen zeggen op the threads die u wel volgt helemaal geen reactie nodig. Met name ontwikkelingsforums worden gedomineerd door drie soorten berichten:

1. berichten met triviale voorstellen;
2. berichten waarin bijval of verschil van mening wordt geuit op iets dat eerder is gezegd; en
3. samenvattende berichten.

Op geen van deze berichten hoeft in principe te worden gereageerd, met name als u er op basis van de voorgaande berichten in de thread redelijk zeker van kunt zijn dat iemand anders gaat zeggen wat u ook zou zeggen. (Wees niet bang dat u met z'n allen in een vicieuze wachtcirkel terecht komt omdat iedereen dezelfde tactiek gebruikt. Er is altijd wel *iemand* die zich in het strijdgewoel stort.) Wanneer u reageert, moet u daar een bepaald doel mee hebben. Vraag uzelf eerst af: weet ik wat

ik wil bereiken? En daarna: wordt dit niet ook bereikt als ik niet reageer?

Twee goede redenen om uw stem in een thread te laten horen zijn a) als u een fout ziet in een voorstel en vermoedt dat u de enige bent die dit gezien heeft en b) als u ziet dat er sprake is van miscommunicatie tussen anderen en u weet dat u dit kunt oplossen door een verklarende post. Het is over het algemeen ook geen probleem om alleen iemand te bedanken voor iets, of "Ik ook!" te zeggen, omdat een lezer direct kan zien dat er op zo'n post geen reactie of verdere actie vereist is en de geestelijke inspanning die vereist is voor de post zodoende volledig eindigt wanneer de lezer de laatste regel van de e-mail bereikt heeft. Maar zelfs dan moet u goed nadenken voordat u iets zegt. Het is altijd beter dat mensen hopen op meer posts van u dan op minder. (Zie de tweede helft van Bijlage C*, Waarom moet ik me druk maken over de kleur van het fietsenhok?* voor meer ideeën over hoe u zich kunt gedragen op een drukke mailinglijst.)

Productieve versus niet-productieve threads

Op een drukke mailinglijst staat u twee dingen te doen. Het eerste is, uiteraard, dat u uit moet zien te vinden waar u aandacht aan gaat besteden en wat u kunt negeren. Het tweede is dat u geen ruis *veroorzaakt*: u wilt niet alleen dat uw posts een hoog opmerkingsgehalte hebben, u wilt ook dat uw posts *andere* mensen stimuleren om posts in te sturen met hetzelfde opmerkingsgehalte, of anders om helemaal geen posts in te sturen.

Om te zien hoe u dat kunt bereiken, kijken we naar de context. Wat zijn kenmerken van een niet-productieve thread?

- Argumenten die eerder al zijn geuit, worden herhaald alsof degene die ze post, denkt dat niemand ze de eerste keer heeft opgemerkt.
- Er wordt steeds meer overdreven en steeds meer mensen raken betrokken, terwijl de belangen steeds kleiner worden.
- Het grootste deel van de reacties komt van mensen die over het algemeen weinig tot niets doen, terwijl mensen die wel wat doen meestal weinig laten horen.
- Veel ideeën worden besproken zonder dat er ooit een duidelijk voorstel wordt gedaan. (Natuurlijk begint ieder interessant idee met een grove notie. De belangrijke vraag is welke richting het vanaf dat moment op gaat. Lijkt het erop dat de thread dit inzicht omzet in iets concreters of mondt het uit in nog meer subinzichten, neveninzichten en discussies over de zin van leven?)

Een thread is niet per definitie tijdverspilling als deze niet vanaf het eerste begin productief is. Hij kan over een erg belangrijk onderwerp gaan (in welk geval het feit dat het nergens naartoe gaat nog meer reden tot bezorgdheid is).

Een thread in een productieve richting sturen zonder opdringerig te zijn, is een kunst op zich. Het werkt niet als u mensen aanspoort geen tijd te verspillen aan trivialiteiten of hun te vragen om alleen een bericht te posten als ze een constructieve inbreng hebben. U kunt dit natuurlijk wel denken, maar als u dit hardop zegt, kunt u mensen beledigen. In plaats daarvan geeft u suggesties over welke kant de thread op zou moeten gaan, door mensen een richting te geven, een route die leidt naar de resultaten waarop u zit te wachten, maar wel zonder dat het klinkt alsof u mensen de les leest over hoe ze zich moeten gedragen. Het verschil tussen de twee zit 'm vooral in de toon. Dit is bijvoorbeeld slecht:

Deze discussie leidt tot niets. Kunnen we dit onderwerp alsjeblieft laten vallen totdat iemand een patch heeft om één van de voorstellen te implementeren? Het heeft geen zin om steeds maar rondjes te blijven draaien en hetzelfde te herhalen. Code zegt meer dan duizend woorden, mensen.

En dit is dan een goed voorbeeld:

Er zijn verschillende voorstellen in omloop in deze thread, maar in geen ervan zijn alle details uitgewerkt, in ieder geval niet genoeg voor een duidelijke stemming voor of tegen. Ook wordt er de laatste tijd niets nieuws meer ingebracht; we herhalen alleen maar wat er eerder ook al is gezegd. Op dit moment zou ik zeggen dat het het beste zou zijn wanneer toekomstige posts óf een volledige specificatie geven van het voorstel óf een patch. Dan hebben we tenminste iets concreets om mee te werken (bijv. het eens worden over de specificatie of de patch toepassen en testen).

Vergelijk de tweede benadering eens met de eerste. De tweede maakt geen onderscheid tussen u en de anderen en beschuldigt anderen er niet van dat zij de discussie in een cirkel laten lopen. Er wordt over 'wij' gesproken, wat erg belangrijk is, of u nou wel of niet eerder hebt deelgenomen aan de thread, omdat het iedereen eraan herinnert dat zelfs degenen die tot nu toe niet veel van zich hebben laten horen belang hebben bij de uitkomst van de thread. Het beschrijft waarom de thread geen enkele kant opgaat, maar zegt dit zonder negatieve bijklank of oordeel; het benoemt alleen zakelijk enkele feiten. En wat het belangrijkst is, het laat een positieve manier van werken zien, waardoor mensen niet het gevoel hebben dat de discussie gesloten is (een restrictie die alleen maar opstandigheid kan uitlokken) maar dat er een manier wordt voorgesteld om de conversatie naar een meer productief niveau te tillen. Dat is een norm waaraan mensen van nature graag willen voldoen.

U hoeft er niet altijd voor te zorgen dat de thread constructiever wordt. Soms wilt u alleen maar dat hij vanzelf verdwijnt. Het doel van uw post is dan de thread de ene of de andere kant op te sturen. Wanneer u uit het verloop van de thread kunt opmaken dat niemand werkelijk de stappen zal nemen die u voorstelt, dan wordt door uw post de thread afgesloten zonder dat die indruk wordt gewekt. Natuurlijk bestaat er geen waterdichte manier om een thread te sluiten, maar zelfs als die er wel was, zou u die niet willen gebruiken. Deelnemers echter vragen om voor duidelijke vooruitgang te zorgen of anders te stoppen met posten, is absoluut gerechtvaardigd, mits het op diplomatieke manier gebeurt. Pas alleen op dat u threads niet te vroeg afbreekt. Een beetje speculatief gebabbel kan productief zijn, al naar gelang van het onderwerp, en als u dan vraagt hiermee te stoppen, kan het creatieve proces in de kiem worden gesmoord en zorgt u er tegelijkertijd voor dat u ongeduldig overkomt.

Verwacht niet dat een thread van het ene op het andere moment stopt. Er volgen

waarschijnlijk nog een paar posts na die van u, of omdat e-mails elkaar kruisen of omdat men het laatste woord wil hebben. Daarover hoeft u zich geen zorgen te maken. Ook hoeft u geen nieuw bericht te posten. Laat de thread vanzelf doodbloeden, of niet doodbloeden, want dat kan ook het geval zijn. U heeft nooit volledige controle over andere mensen, maar aan de andere kant kunt u er wel vanuit gaan dat u in vele threads statistisch gezien een aanzienlijke invloed hebt.

Hoe makkelijker het onderwerp, des te langer de discussie

Hoewel discussies binnen ieder onderwerp van de rode draad kunnen gaan afwijken, is de kans hierop groter wanneer de technische complexiteit van het onderwerp minder wordt. Tenslotte kunnen minder deelnemers volgen wat er gebeurt als een onderwerp technisch erg moeilijk is. Degenen die dat wel kunnen, zijn hoogstwaarschijnlijk de meest ervaren ontwikkelaars, die al duizenden keren dergelijke discussies hebben gevoerd en weten welk soort gedrag leidt tot de consensus waarmee iedereen kan leven.

Daarom is het het moeilijkst om consensus bereiken over technische vragen die makkelijk te begrijpen zijn en waarover iedereen een mening heeft en over 'softe' onderwerpen, zoals organisatie, publiciteit, financiering enz. Mensen kunnen eindeloos in dergelijke discussies blijven hangen, omdat men niet gekwalificeerd hoeft te zijn om mee te mogen praten, omdat er (zelfs achteraf) geen duidelijke manier is om vast te stellen of een beslissing goed of fout is en omdat het soms een goede tactiek kan zijn de reacties van andere deelnemers in de discussie af te wachten.

Het principe dat de hoeveelheid discussie omgekeerd evenredig is aan de complexiteit van het onderwerp is al lang bekend en staat informeel ook bekend als het *fietsenhokeffect*. Poul-Henning Kamp gaf hiervoor in een inmiddels beroemde post aan BSD-ontwikkelaars een verklaring:

Het is een lang verhaal, of eigenlijk een heel oud verhaal, en eigenlijk toch een kort verhaal. C. Northcote Parkinson schreef in het begin van de jaren 60 van de vorige eeuw een boek met de titel 'De wet van Parkinson', met veel inzichten in de dynamiek van management.

[...]

In het specifieke voorbeeld van het fietsenhok is de andere essentiële component een kerncentrale. Dat geeft denk ik een goede datering van dit boek.

Parkinson laat zien hoe u in de raad van bestuur goedkeuring kunt krijgen voor de bouw van een kerncentrale van miljoenen of zelfs miljarden dollars, maar dat u in eindeloze discussies verzeild raakt als u een fietsenhok wilt neerzetten.

Parkinson legt uit dat dit komt doordat een kerncentrale zo enorm groot, duur en gecompliceerd is dat mensen er met hun hoofd niet bij kunnen. In plaats van te proberen het toch te begrijpen, vertrouwen ze op de aanname dat iemand anders de informatie in een vorig stadium wel gecontroleerd zal hebben. Richard P. Feynmann geeft in zijn boeken een paar interessante en zeer treffende voorbeelden met betrekking tot Los Alamos.

Aan de andere kant hebben we het fietsenhok. Iedereen kan in een weekend een fietsenhok bouwen en dan nog genoeg tijd over houden om op zondag naar Studio Sport te kijken. Het maakt niet uit hoe goed voorbereid u bent en hoe redelijk uw voorstel is, iemand zal altijd de kans grijpen om te laten zien dat hij zijn werk doet, dat hij goed oplet, dat hij *er is*.

We noemen dit 'je geur achterlaten'. Het heeft te maken met persoonlijke trots en aanzien; je moet iets aan kunnen wijzen en zeggen "Kijk! Dat heb *ik* gedaan." Politici bijvoorbeeld hebben sterk die neiging, maar iedereen loopt kans ermee besmet te raken. Denk maar aan het achterlaten van voetstappen in nat cement.

(Zijn volledige post is de moeite van het lezen waard. Zie Bijlage C, *Waarom zou ik me druk maken over de kleur van het fietsenhok?* Of http://bikeshed.com.)

Iedereen die wel eens te maken heeft gehad met het nemen van groepsbeslissingen weet waar Kamp het over heeft. Het is over het algemeen echter onmogelijk om *iedereen* ervan te overtuigen dat ze het fietsenhok niet hoeven schilderen. Het beste dat u kunt bereiken, wanneer u het ziet gebeuren, is laten zien dat het fenomeen bestaat en de seniorontwikkelaars (de mensen die het meest in de melk te brokkelen hebben) hun verfkwasten zo snel mogelijk weg te laten leggen zodat zij in ieder geval niet bijdragen aan de ruis. Deze fietsenhokdiscussies zullen nooit helemaal verdwijnen, maar door mensen binnen het project over het fenomeen te informeren kunt u zorgen dat ze korter worden en minder vaak voorkomen.

Voorkom heilige oorlogen

Een *heilige oorlog* is een discussie, vaak - maar niet altijd - over een relatief onbelangrijke kwestie, die niet kan worden opgelost op basis van argumenten, maar waarbij mensen zo emotioneel betrokken zijn, dat ze blijven discussiëren in de hoop gelijk te krijgen. Een heilige oorlog is niet hetzelfde als een fietsenhokdiscussie. Mensen die fietsenhokken verven, hebben meestal snel hun mening klaar (omdat ze die hebben) maar zijn er niet per se emotioneel bij betrokken en kunnen ook andere tegenstrijdige argumenten aandragen om aan te geven dat ze alle gezichtspunten van de kwestie begrijpen. In een heilige oorlog is begrip voor andere argumenten echter een teken van zwakheid. In een heilige oorlog is iedereen het erover eens dat er maar één antwoord juist is; men verschilt echter van mening over welk antwoord dat is.

Zodra een heilige oorlog is begonnen kan deze meestal niet naar ieders tevredenheid worden opgelost. U doet er geen goed aan door te zeggen dat er sprake is van een heilige oorlog terwijl die volop aan de gang is. Dat weet iedereen namelijk al. Helaas is een algemeen kenmerk van heilige oorlogen dat er geen overeenstemming bestaat over de vraag of het geschilpunt sowieso wel met discussiëren kan worden opgelost. Van een afstand bezien is duidelijk dat geen van beide kanten de mening van de ander kan veranderen. Maar de deelnemers zelf denken dat de andere partij gewoon wat traag van begrip is en niet helder denkt, en dat hij wel overstag gaat als hij maar genoeg met argumenten wordt bestookt. Ik zeg *niet* dat

er nooit iemand gelijk heeft in een heilige oorlog. Soms is dat wel het geval. Bij de heilige oorlogen waaraan ik deel heb genomen, was ik altijd degene die gelijk had, uiteraard. Maar dat maakt helemaal niet uit, er bestaat namelijk geen formule voor om overtuigend aan te tonen dat één van beide partijen gelijk heeft.

Een algemene, maar weinig voldoening biedende manier om een heilige oorlog op te lossen is door te zeggen “We hebben al veel meer tijd en energie aan de discussie besteed dan het onderwerp waard is! Kunnen we het niet gewoon laten voor wat het is?" Deze oplossing kent twee problemen. Het eerste is dat er al tijd en energie besteed zijn en dat die nooit meer herwonnen kunnen worden. De enige vraag die overblijft, is hoe veel energie er *nog meer* in moet worden gestoken. Wanneer sommige mensen geloven dat een kleine beetje extra discussie de kwestie kan oplossen, dan heeft het (voor hen) nog steeds zin om ermee door te gaan.

Het andere probleem is dat wanneer wordt gevraagd om de kwestie te laten vallen dit vaak gelijkstaat aan het tot winnaar uitroepen van de partij die voor de status quo staat, omdat er niets verandert. En in sommige gevallen is het behouden van de status quo ook niet acceptabel: iedereen is het erover eens dat er een beslissing moet worden genomen en dat er tot actie moet worden overgegaan. De kwestie laten vallen is voor iedereen een nog slechtere optie dan een eigen argument moeten opgeven. Maar omdat dit dilemma voor iedereen in gelijke mate geldt, is het nog steeds mogelijk om eindeloos door te blijven discussiëren over wat er moet gebeuren.

Dus hoe moet u met heilige oorlogen omgaan?

Het eerste antwoord op deze vraag is: de zaken zo opzetten, dat ze niet voorkomen. Dit is niet zo’n hopeloze taak als het lijkt.

Op bepaalde standaard heilige oorlogen kunt u anticiperen: ze gaan vaker over de programmeertaal, licenties (zie het gedeelte ‘De GPL en compatibiliteit van licenties’ in Hoofdstuk 9, *Licenties, auteursrechten en patenten*), doordraven over ‘Reply to’ (zie het gedeelte ‘Het grote ‘Reply to’-debat’ in Hoofdstuk 3, T*echnische infrastructuur*) en nog een paar van zulke onderwerpen. Meestal heeft ieder project zo wel een heilig oorlogje of twee. Ervaren ontwikkelaars raken hier snel genoeg aan gewend. De technieken voor het stoppen van heilige oorlogen, of in ieder geval voor het beperken van de schade ervan, zijn overal ongeveer gelijk. Zelfs als u zeker weet dat u gelijk hebt, probeer dan in ieder geval *een beetje* sympathie en begrip te tonen voor de argumenten van de tegenpartij. Vaak is het grootste probleem van een heilige oorlog dat beide zijden een zo hoog mogelijke muur hebben opgetrokken en duidelijk hebben gemaakt dat iedere afwijkende mening je reinste onzin is. Daardoor is overgave of van gedachten veranderen psychologisch ondraaglijk geworden, Het betekent niet alleen toegeven dat je ongelijk had, maar dat je iets *zeker* wist en ongelijk had. De manier om dit aanvaardbaar te maken voor de andere partij is door zelf aan te geven onzeker te zijn, juist door te laten zien dat u hun argumenten begrijpt en ze logisch vindt, misschien zelfs overtuigend. Maak een gebaar dat ruimte biedt voor een soortgelijk gebaar van de andere kant. Gewoonlijk verbetert de situatie dan wel. De kans dat u het technische resultaat krijgt dat u wilde, wordt er niet groter of kleiner van, maar u kunt in ieder geval onnodige gevolgschade aan het moreel van het project voorkomen.

Wanneer een heilige oorlog niet kan worden voorkomen, beslis dan meteen aan het begin hoeveel het u eigenlijk kan schelen en wees bereid openlijk toe te geven. Wanneer u dat doet, kunt u zeggen dat u zich terugtrekt omdat u het geen heilige oorlog waard vindt. Toon echter geen bitterheid en gebruik de gelegenheid *niet* om nog een laatste hatelijke opmerking te maken over de argumenten van de tegenpartij. Opgeven is alleen effectief als het met stijl wordt gedaan.

Heilige oorlogen over de programmeertaal zijn een geval apart, omdat ze vaak erg technisch van aard zijn, terwijl er toch veel mensen zijn die zich gekwalificeerd genoeg voelen om er aan deel te nemen. Ook staan er veel belangen op het spel, omdat het resultaat van de oorlog bepaalt in welke taal een groot deel van de code van het project wordt geschreven. De beste oplossing hiervoor is om de taal in een vroeg stadium te kiezen, met behulp van invloedrijke ontwikkelaars die er vanaf het begin bij zijn betrokken en de keuze verdedigen door te zeggen dat het de taal is waar iedereen zich prettig bij voelt, *niet* omdat hij beter is dan andere talen die in plaats daarvan ook hadden kunnen worden gebruikt. Laat een discussie nooit ontaarden in een academische vergelijking van programmeertalen (dit lijkt extra vaak te gebeuren wanneer iemand met Perl op de proppen komt, om wat voor reden dan ook). Dit is een gesloten hoofdstuk waarbij u gewoon moet weigeren betrokken te raken.

Voor een meer historische achtergrond over heilige oorlogen, zie http://catb.org/~esr/jargon/html/H/holy-wars.html en het artikel van Danny Cohen waarmee de term populair werd http://www.ietf.org/rfc/ien/ien137.txt.

Het effect van de ‘luidruchtige minderheid’

Tijdens discussies op mailinglijsten gebeurt het maar al te vaak dat een kleine minderheid, door de lijst met talrijke en lange e-mails te bestoken, de indruk wekt dat een groot deel van de lijst het ergens niet mee eens is. Het lijkt een beetje op obstructie in het politieke debat, behalve dat de illusie van een wijdverbreide afwijkende mening nog sterker is omdat die wordt geuit in een willekeurig aantal afzonderlijke posts en de meeste mensen niet de moeite nemen bij te houden wie wat heeft gezegd en wanneer. Men krijgt echter instinctief de indruk dat het onderwerp erg controversieel is en wil wachten totdat de storm is geluwd.

De beste manier om dit effect tegen te gaan is door een en ander zeer duidelijk uit te leggen en met bewijs te komen voor hoe klein het werkelijke aantal opposanten is vergeleken met het aantal mensen dat het er wel mee eens is. Om het verschil tussen voor- en tegenstanders nog groter te maken kunt u privé navraag doen bij mensen die meestal hun mond houden maar waarvan u denkt dat ze het met de meerderheid eens zijn. Zeg nooit iets waarmee u kunt suggereren dat de opposanten doelbewust proberen de boel op te blazen. De kans is groot dat dit helemaal niet het geval is, en zelfs als dat wel het geval zou zijn, heeft het geen enkel strategisch voordeel dit hardop te zeggen. Het enige dat u hoeft doen, is de aantallen naast elkaar te zetten zodat mensen zich realiseren dat hun intuïtie over de situatie niet overeenkomt met de werkelijkheid.

Dit advies is niet van toepassing op kwesties met duidelijke voor- en tegenargumenten. Het is van toepassing op discussies waar veel drukte over wordt gemaakt, maar waarvan niet duidelijk is of de meesten wel geloven dat de kwestie een reëel probleem is. Na een poosje kunt u, wanneer u het erover eens bent dat het niet de moeite waard is om actie over de kwestie te nemen en u kunt zien dat deze niet veel garen spint (ook al worden er wel veel e-mails gegenereerd) publiekelijk opmerken dat er niet voldoende garen gesponnen wordt. Als er sprake was van het 'luidruchtige minderheid'-effect zal uw post op de mensen overkomen als een frisse wind door de gelederen. De indruk die de meeste mensen van de discussie tot dan toe hebben, zal weinig verheffend zijn: "Hm, het lijkt echt wel alsof er iets belangrijks aan de hand is, want er zijn ontzettend veel posts, maar ik zie geen duidelijke vooruitgang." Door uit te leggen hoe de vorm van de discussie ervoor zorgde dat het turbulenter leek dan het in feite was, krijgen mensen een nieuw kader waardoor ze hun inzichten over wat er zich afspeelde kunnen herzien.

6.3 MOEILIJKE MENSEN

Moeilijke mensen zijn binnen elektronische platforms net zo moeilijk in de omgang als daarbuiten. Met 'moeilijk' bedoel ik niet 'onbeleefd'. Onbeleefde mensen zijn irritant, maar ze zijn niet per definitie moeilijk. Ik heb eerder in dit boek al besproken hoe u met hen kunt omgaan. Reageer meteen de eerste keer op de onbeleefdheden en negeer ze daarna of behandel ze net als ieder ander dat doet. Wanneer ze onbeleefd blijven, maken ze zich meestal niet populair bij de rest. Daardoor is de kans groot dat ze weinig invloed krijgen op het project. Het is eigenlijk een probleem dat zichzelf oplost.

De echt moeilijke gevallen zijn mensen die niet openlijk onbeleefd zijn maar die de projectprocessen manipuleren of saboteren waardoor andere mensen tijd en energie verliezen zonder dat het project daar baat bij heeft. Dergelijke mensen zoeken vaak naar mazen in de procedures van het project om zichzelf meer invloed te geven dan ze anders misschien zouden hebben. Dit is veel verraderlijker dan pure onbeleefdheid, omdat het gedrag en de daaruit voortvloeiende schade geen van beide duidelijk zichtbaar zijn voor de toevallige voorbijganger. Een klassiek voorbeeld is de filibuster, iemand die een politiek debat vertraagt door eindeloze redevoeringen, waarbij iemand (die natuurlijk altijd zo verstandig overkomt als maar mogelijk is) blijft zeggen dat de besproken kwestie nog niet toe is aan een oplossing en steeds meer mogelijke oplossingen of nieuwe gezichtspunten voor oude oplossingen blijft aandragen, terwijl hij in feite aanvoelt dat het project op het punt staat om consensus te bereiken over de kwestie of om het ter stemming voor te leggen en hij het niet eens is met de waarschijnlijke uitkomst. Een ander voorbeeld is wanneer er een debat plaatsvindt dat maar niet tot consensus lijkt te kunnen komen, maar waarbij de groep in ieder geval probeert de punten van onenigheid te verduidelijken en een samenvatting te maken waaraan iedereen kan refereren. De dwarsligger, die weet dat de samenvatting kan leiden tot een oplossing waar hij het niet mee eens is, probeert vaak zelfs de samenvatting te vertragen, door halsstarrig de vraag wat er wel en niet in moet gecompliceerder te maken, door het niet eens te zijn met redelijke suggesties of door onverwachte nieuwe aspecten in te brengen.

Omgaan met moeilijke mensen

Om dit soort gedrag tegen te kunnen gaan, is het nuttig om de mentaliteit van dergelijke mensen te begrijpen. Mensen doen over het algemeen niet bewust moeilijk. Niemand wordt 's morgens wakker en zegt tegen zichzelf: "Vandaag zal ik eens op cynische wijze proberen de procedures te manipuleren en me als irritante dwarsligger gaan gedragen." In plaats daarvan ligt aan dergelijk gedrag vaak een semiparanoïde gevoel ten grondslag, dat iemand van de interacties en de beslissingen binnen de groep wordt uitgesloten. De persoon heeft het gevoel dat hij niet serieus wordt genomen of (in meer ernstige gevallen) dat er bijna sprake is van een samenzwering tegen hem, dat de andere projectleden hebben besloten een exclusief clubje te vormen waarvan hij geen deel uitmaakt. Voor hem rechtvaardigt dit de keuze om de regels letterlijk op te vatten en de procedures van het project te manipuleren, om er zo voor te zorgen dat iedereen hem *wel* serieus neemt. In extreme gevallen gelooft de persoon dat hij een eenzame strijd voert om het project te redden.

Het ligt in de aard van dergelijke bedreigingen van binnenuit, dat niet iedereen het verschijnsel tegelijkertijd in de gaten heeft en sommigen het helemaal niet zien, totdat ze met de harde feiten worden geconfronteerd. Dit betekent dat het erg moeilijk kan zijn om dit fenomeen te neutraliseren. Het is niet voldoende als u zichzelf ervan overtuigt dat dit gebeurt. U moet voldoende bewijsmateriaal verzamelen om ook anderen hiervan te overtuigen en u moet dit bewijsmateriaal op doordachte wijze verspreiden.

Omdat het zo moeilijk is om dit probleem tegen te gaan, is het vaak beter om het eerst een poosje te tolereren. Zie het als een parasitaire, maar niet ernstige ziekte. Als deze het project niet te zeer ondermijnt, is het niet erg als de infectie blijft bestaan. En medicijnen kunnen negatieve bijwerkingen hebben. Wanneer het probleem echter onverantwoord veel schade gaat aanrichten, is het tijd om tot actie over te gaan. Begin met het verzamelen van aantekeningen van wat u ziet gebeuren. Zorg ervoor dat u ook referenties naar openbare archieven opneemt. Dit is één van de redenen waarom er een archief wordt bijgehouden van het project, dus kunt u het dan ook maar beter gebruiken. Zodra u sterk staat, kunt u privégesprekken voeren met andere deelnemers aan het project. Vertel ze niet direct wat u hebt geconstateerd, maar vraag ze eerst wat zij hebben gezien. Dit is waarschijnlijk uw laatste kans op ongecensureerde feedback over hoe anderen het gedrag van de onruststoker zien. Zodra u begint er openlijk over te praten, zullen de meningen zich polariseren en zal niemand zich nog kunnen herinneren hoe ze voorheen over de zaak dachten.

Als uit de privégesprekken blijkt dat minstens een paar anderen het probleem ook zien, is het tijd iets te ondernemen. Vanaf dat moment moet u *heel* voorzichtig te werk gaan. Het is voor dit soort mensen namelijk zeer eenvoudig om het te laten voorkomen alsof u ten onrechte op ze afgeeft. Wat u ook doet, beschuldig iemand er nooit van willens en wetens de procedures van het project te misbruiken, paranoide te zijn of, in algemene zin, van al het andere waarvan u sterke vermoedens hebt. Uw strategie zou moeten zijn om altijd verstandiger en meer gericht op het algemene belang van het project over te komen dan de ander, met als doel het gedrag van die persoon te veranderen of hem het project permanent te laten verlaten. Afhan-

kelijk van de andere ontwikkelaars en uw relatie met hen kan het handig zijn eerst privé medestanders te verzamelen. Het effect hiervan kan echter ook averechts zijn. Het veroorzaakt achter de schermen namelijk alleen maar onrust wanneer mensen denken dat u een misplaatste roddelcampagne bent gestart.

Vergeet niet dat, hoewel de andere persoon degene is die destructief gedrag vertoont, *u* dan degene bent die destructief overkomt wanneer u uw publieke beschuldigingen niet hard kunt maken. Zorg ervoor dat u voldoende voorbeelden heeft waarmee u kunt laten zien wat u bedoelt en zeg het zo beleefd mogelijk, maar wel direct. Misschien bent u niet in staat de betreffende persoon te overtuigen, maar het is voldoende als u alle anderen weet te overtuigen.

Casestudy

Ik herinner me uit mijn ruim tien jaar ervaring met open source-softwareprojecten slechts één situatie waarbij het zo uit de hand liep dat we iemand moesten vragen helemaal te stoppen met het sturen van posts. Zoals zo vaak was deze meneer niet onbeleefd en wilde hij oprecht behulpzaam zijn. Hij wist alleen niet wanneer hij wel en wanneer hij niet moest posten. Onze mailinglijsten waren publiekelijk toegankelijk en hij postte zo veel en stelde zo veel vragen over zo veel verschillende onderwerpen dat het een probleem werd binnen de gemeenschap. We hadden al eens geprobeerd hem op een nette manier te vragen wat meer vooronderzoek te doen voordat hij een post verstuurde, maar dat had geen effect.

De strategie die uiteindelijk werkte, was een perfect voorbeeld van hoe u sterk kunt staan op basis van neutrale en kwantitatieve informatie. Eén van onze ontwikkelaars had wat speurwerk gedaan in de archieven en stuurde het volgende bericht privé naar enkele andere ontwikkelaars. De boosdoener (de derde naam op de lijst, hier 'J. Random') had nog weinig met het project van doen gehad en had geen code of documentatie geschreven. Toch was hij de op twee na meest actieve poster op de mailinglijst:

```
Van: "Brian W. Fitzpatrick" <fitz@collab.net>
Aan: [... ontvangerslijst weggelaten i.v.m. anonimiteit ...]
Onderwerp: De bodemloze energieput van Subversion
Datum: Woensdag 12 Nov 2003 23:37:47 -0600

De afgelopen 25 dagen zijn de onderstaande 6 personen de mensen
met de meeste posts naar de svn [dev|users]:

    294  kfogel@collab.net
    236  "C. Michael Pilato" <cmpilato@collab.net>
    220  "J. Random" <jrandom@problematic-poster.com>
    176  Branko Čibej <brane@xbc.nu>
    130  Philip Martin <philip@codematters.co.uk>
    126  Ben Collins-Sussman <sussman@collab.net>

Ik zou zeggen dat vijf van deze mensen een bijdrage leveren aan
de release van Subversion 1.0 in de nabije toekomst.

Ik zou ook zeggen dat één van deze personen constant tijd en
energie in beslag neemt van de andere 5, om maar niet te spreken
van de andere deelnemers van de lijst, waardoor hij (zij het on-
bedoeld) de ontwikkeling van Subversion vertraagt.  Ik heb geen
grondige analyse gedaan, maar als ik het archief van Subversion
bekijk zie ik dat iedere mail van deze persoon minimaal één keer
wordt beantwoord door minstens 2 van de andere 5 personen uit de
bovenstaande lijst.

Ik denk dat er radicaal moet worden ingegrepen, zelfs als dit
betekent dat we de bovengenoemde persoon wegjagen.  Van genuan-
ceerde opmerkingen en vriendelijkheid is al gebleken dat ze geen
effect hebben.

dev@subversion is een mailinglijst ter ondersteuning van de ont-
wikkeling van een versiebeheersysteem, geen groepstherapie.

-Fitz, in een poging zich door drie dagen opgestapelde svn-mail
heen te werken
```

Hoewel dit op het eerste gezicht misschien niet zo lijkt, was het gedrag van J. Random een klassiek geval van het misbruiken van de projectprocedures. Hij deed niet iets waar je onmiddellijk bezwaar tegen zou maken, zoals een stemming proberen te saboteren, maar hij maakte misbruik van het mailinglijstbeleid, dat was gebaseerd of zelfrestrictie van de leden ten aanzien van hun posts. We hadden het aan ieders beoordelingsvermogen overgelaten wanneer men postte en over welke onderwerpen. Daardoor hadden we geen procedures om op terug te vallen toen er sprake was van iemand die niet beschikte over een dergelijk beoordelingsvermogen of weigerde dit te gebruiken. Er waren geen regels waarvan we konden zeggen dat de man ze aan het overtreden was, maar iedereen wist dat zijn veelvuldige posts een serieus probleem begonnen te worden.

De strategie van Fitz was, achteraf gezien, meesterlijk. Hij verzamelde kwantitatief belastend bewijsmateriaal en verspreide dit vervolgens op discrete wijze, door het eerst naar een paar mensen te sturen wiens steun belangrijk zou zijn bij eventuele drastische maatregelen. Ze kwamen overeen dat er in ieder geval iets moest worden ondernomen en uiteindelijk hebben we J. Random opgebeld, het probleem direct aan hem voorgelegd en hem gevraagd te stoppen met posten. Hij heeft nooit werkelijk begrepen waarom dit was. Als hij in staat zou zijn geweest dit te begrijpen, was hij waarschijnlijk nooit begonnen met zijn obstructiegedrag. Maar hij ging ermee akkoord om niet meer te posten en de mailinglijst werd weer werkbaar. Deze strategie werkte deels misschien door de impliciete dreiging dat we zijn posts zouden kunnen weigeren met behulp van moderatorsoftware die normaal gesproken wordt gebruikt om spam tegen te houden (zie het gedeelte 'Spam voorkomen' in Hoofdstuk 3, *Technische infrastructuur*). Maar de belangrijkste reden dat we deze optie achter de hand hadden, was dat Fitz de noodzakelijke steun had ingewonnen van de sleutelfiguren binnen het project.

6.4 OMGAAN MET GROEI

De prijs van succes is hoog in het wereldje van open source-software. Zodra uw software populairder wordt, neemt het aantal mensen dat opduikt op zoek naar informatie drastisch toe, terwijl het aantal mensen dat in staat is informatie te verstrekken veel langzamer groeit. Maar zelfs wanneer deze verhouding wel gelijk zou zijn, bestaat er een fundamenteel uitbreidbaarheidsprobleem door de manier waarop de meeste open source-projecten met communicatie omgaan. Laten we bijvoorbeeld eens naar mailinglijsten kijken. De meeste projecten hebben mailinglijsten voor algemene vragen van gebruikers. Soms heet de lijst 'users', 'discuss', 'help' of iets dergelijks. Maar wat de naam ook is, het doel van de lijst is altijd hetzelfde: een omgeving bieden waar mensen vragen krijgen op hun antwoorden, terwijl anderen toekijken en (vermoedelijk) kennis in zich opnemen door deze vragen en antwoorden te observeren.

Deze mailinglijsten werken prima tot een paar duizend gebruikers en/of een paar honderd posts per dag. Maar ongeveer als dat aantal bereikt is, begint het systeem te wankelen, omdat iedere deelnemer iedere post te zien krijgt. Wanneer het aantal posts groter wordt dan het aantal dat iedere lezer per dag aankan, dan wordt de lijst een belasting voor de leden. Stelt u zich eens voor dat Microsoft een dergelijke mailinglijst zou hebben voor Windows XP. Windows XP heeft honderden miljoenen gebruikers. Zelfs als ééntiende of zelfs maar één procent daarvan elk etmaal een vraag zou stellen, dan zou deze hypothetische lijst honderdduizenden posts per dag krijgen! Zo'n lijst is natuurlijk nooit levensvatbaar, gewoon omdat niemand er lid van blijft. Dit probleem beperkt zich niet alleen tot mailinglijsten. Hetzelfde geldt voor IRC-kanalen, online discussieforums en ieder ander systeem waarbij een groep de vragen van individuen te zien krijgt. De implicaties daarvan zijn schokkend: het gebruikelijke open source-model van immense parallelle ondersteuning laat zich gewoonweg niet opschalen naar het niveau dat nodig is om de wereld te veroveren.

Er is geen explosie te zien wanneer een forum zijn breekpunt bereikt. Zichtbaar is alleen het negatieve feedbackeffect: mensen schrijven zich uit van de mailinglijst, gaan weg bij het IRC-kanaal of stoppen met het stellen van vragen omdat ze merken dat ze tussen alle andere posts niet worden gehoord. Als steeds meer mensen deze uiterst rationele beslissing nemen, lijkt het alsof de activiteit van het forum gemakkelijk bestuurbaar blijft. Maar het blijft bestuurbaar omdat de verstandige (of in ieder geval ervaren) mensen ergens anders op zoek zijn gegaan naar informatie, terwijl de onervaren mensen achterblijven en blijven posten. Met andere woorden, een neveneffect van het blijven gebruiken van niet-uitbreidbare communicatiemodellen wanneer het project groeit, is dat de gemiddelde kwaliteit van zowel de vragen als de antwoorden omlaag gaat. Daardoor lijkt het alsof nieuwe gebruikers dommer zijn dan voorheen, terwijl dat waarschijnlijk in werkelijkheid niet het geval is. Het belangrijkste is dat het rendement van het gebruik van dergelijke overvolle forums achteruit gaat, waardoor het heel logisch is dat mensen met ervaring eerst ergens anders op zoek gaan naar antwoorden. Het aanpassen van de communicatiemechanismen aan de groei van het project vereist dan ook twee aan elkaar gerelateerde strategieën:

1. Herkennen wanneer bepaalde delen van een forum *niet* te lijden hebben onder de ongebreidelde groei, zelfs wanneer dat voor het forum in zijn geheel wel het geval is, en deze delen afsplitsen naar nieuwe, meer gespecialiseerde forums (d.w.z., gooi het kind niet met het badwater weg).
2. Ervoor zorgen dat er veel geautomatiseerde informatiebronnen beschikbaar zijn en dat deze goed georganiseerd, up-to-date en gemakkelijk te vinden zijn.

Strategie (1) is meestal niet zo moeilijk. De meeste projecten beginnen met één algemeen forum: een algemene discussielijst, waarop iedereen terecht kan met ideeën voor functies, ontwerpvragen en problemen met de code. Iedereen die bij het project betrokken is, is ook lid van de lijst. Na een poosje wordt het meestal duidelijk dat de lijst zich heeft ontwikkeld in een aantal sublijsten met afzonderlijke onderwerpen. Sommige threads gaan duidelijk over ontwikkeling en ontwerp, anderen zijn gebruikersvragen als "Hoe moet ik X doen?" en misschien is er een derde groep onderwerpen rond het afhandelen van bugrapporten en verbeteringsverzoeken enz. Ieder individu kan natuurlijk deelnemen in vele verschillende threads, maar het belangrijkste is dat er niet veel overlap is tussen de verschillende type threads. Ze kunnen worden opgesplitst in afzonderlijke lijsten zonder dat er schadelijke wrijving kan ontstaan, omdat threads zelden over de grenzen van de onderwerpen heen gaan.

In feite is deze opsplitsing een proces met twee fasen. U creëert de nieuwe lijst (of IRC-kanaal, of wat dan ook) en blijft vervolgens mensen net zolang zachtjes aan hun hoofd zeuren totdat ze de nieuwe forums daadwerkelijk correct *gebruiken*. Deze fase kan weken in beslag nemen, maar uiteindelijk zal bij de mensen het kwartje wel vallen. Het enige wat u hoeft doen, is de afzender altijd vertellen wanneer een post naar de verkeerde lijst is gestuurd, en dat ook zichtbaar te doen, zodat andere mensen worden aangemoedigd om te helpen met het routeren. Het kan ook handig zijn om een webpagina te hebben met een richtlijn voor alle beschikbare lijsten. In uw reacties kunt u naar deze webpagina verwijzen waarbij, als extra bonus, de ontvanger ook nog iets kan leren over het zoeken naar richtlijnen voordat hij een post verstuurt.

Strategie (2) is een doorlopend proces, dat tijdens de gehele projectcyclus doorgaat en waar veel deelnemers bij zijn betrokken. Dit is natuurlijk voor een deel een kwestie van het hebben van up-to-date documentatie (zie het gedeelte 'Documentatie' in Hoofdstuk 2, *Aan de slag*) en mensen daar ook naar te verwijzen. Maar er komt nog veel meer bij kijken. De volgende gedeelten geven een gedetailleerde beschrijving van deze strategie.

Opvallend gebruik van archieven

Normaal gesproken wordt alle communicatie (behalve sommige IRC-discussies) gearchiveerd. De archieven zijn voor iedereen toegankelijk, kunnen worden doorzocht en hebben referentiële stabiliteit: dat wil zeggen dat wanneer informatie wordt opgeslagen op een bepaald adres, het altijd op dat adres beschikbaar blijft.

Gebruik deze archieven zo veel mogelijk en zo opvallend mogelijk. Zelfs wanneer

u het antwoord op een vraag uit uw hoofd weet, als u denkt dat er een referentie in de archieven te vinden is dat het antwoord bevat, neem dan de tijd om dit op te zoeken en te laten zien. Iedere keer dat u dit op een voor iedereen zichtbare manier doet, zullen er weer enkele mensen ontdekken dat de archieven bestaan en dat het zoeken in de archieven productieve resultaten kan opleveren. Tevens versterkt u de heersende norm over het dupliceren van informatie wanneer u refereert aan het archief in plaats van het advies te herschrijven. Waarom zouden de antwoorden op twee verschillende plaatsen beschikbaar moeten zijn? Als het aantal plaatsen waarop het antwoord kan worden gevonden tot een minimum wordt beperkt, is de kans groter dat mensen die het hebben gevonden zullen onthouden waar ze naar moeten zoeken om het opnieuw te vinden. Goedgeplaatste referenties dragen ook bij aan de kwaliteit van de zoekresultaten in het algemeen, omdat ze de positie van de bron in de rangorde van internetzoekmachines versterken.

Er zijn echter momenten waarop het dupliceren van informatie wel zinvol is. Neem bijvoorbeeld een antwoord in het archief, dat niet van u is, waarin staat:

```
Het lijkt erop dat er met uw Scanley-indexen 'gefrobd' is.  Om
dit terug te draaien moet u de volgende stappen volgen:

1. Zet de Scanley-server uit.
2. Start het programma 'defrobnicate' van Scanley.
3. Start de server weer op.
```

Vervolgens ziet u maanden later een andere post waarin iemand aangeeft dat zijn indexen zijn gefrobd. U zoekt in de archieven en vindt het bovenstaande antwoord, maar u realiseert zich dat er enkele stappen ontbreken (misschien per ongeluk of misschien omdat de software is veranderd sinds deze post werd geschreven). De klassieke manier om dit op te lossen is het posten van nieuwe, completere instructies en expliciet aangeven dat de oude post verouderd is door het te vermelden:

```
Het lijkt erop dat er met uw Scanley-indexen 'gefrobd' is.  In
juli zijn we dit probleem al eerder tegengekomen en J. Random
poste een oplossing op http://blahblahblah/blah.  Onderstaand
een completere beschrijving over hoe u uw indexen kunt "ontfrob-
ben", gebaseerd op de instructies van J. Random maar met enkele
uitbreidingen:

1. Zet de Scanley-server uit.
2. Log in als de gebruiker waaronder de Scanley-server normaal
   gesproken actief is.
3. Start als deze gebruiker het programma 'defrobnicate' voor de
   indexen.
4. Start Scanley handmatig op om te zien of de indexen nu wel
   werken.
5. Start de server weer op.
```


(In het ideale geval zou het mogelijk moeten zijn om een notitie toe te voegen aan een oude post, met de vermelding dat er nieuwe informatie beschikbaar is en te verwijzen naar de nieuwe post. Ik ken echter geen archiveringssoftware die het mogelijk maakt om een vermelding 'verouderd door' toe te voegen, misschien omdat het een beetje lastig is dit op zo'n manier te implementeren dat het de integriteit van het archief als woordelijk naslagwerk niet aantast. Dit is nog een extra reden waarom het maken van een speciale webpagina met antwoorden op algemene vragen een goed idee is.)

Archieven worden waarschijnlijk het meest afgezocht naar antwoorden op technische vragen, maar het belang ervan voor het project gaat veel verder. De formele richtlijnen van een project kunnen worden gezien als de officiële regels, de archieven als de gewoonteregels: een overzicht van alle genomen beslissingen en hoe tot deze beslissingen werd gekomen. In iedere terugkerende discussie wordt tegenwoordig als min of meer verplicht gezien om te beginnen met een zoektocht door de archieven. Hierdoor kunt u de discussie beginnen met een samenvatting van de huidige stand van zaken, anticiperen op tegenwerpingen, tegenbewijs verzamelen en misschien invalshoeken ontdekken waaraan u nog niet had gedacht. De andere deelnemers zullen ook van u *verwachten* dat u het archief hebt nagezocht. Zelfs als voorgaande discussies nergens toe geleid hebben, moet u er toch naar verwijzen wanneer u het onderwerp opnieuw aansnijdt, zodat mensen zelf kunnen zien dat a) de discussie nergens toe leidde en b) u uw huiswerk hebt gedaan en u daarom waarschijnlijk nu iets zegt dat nog niet eerder is gezegd.

Behandel alle bronnen als archieven

Alle bovenstaande adviezen hebben betrekking op meer dan alleen de archieven van de mailinglijsten. Het zou een organisatorisch principe van het project moeten zijn om bepaalde stukken informatie beschikbaar te hebben op stabiele en makkelijk vindbare adressen. Laten we de FAQ van het project als voorbeeld nemen.

Hoe gebruiken mensen een FAQ?

1. Ze willen erin zoeken naar specifieke woorden of uitdrukkingen.
2. Ze willen er doorheen kunnen bladeren, om informatie te verzamelen zonder dat ze specifiek op zoek zijn naar bepaalde antwoorden.
3. Ze verwachten dat zoekmachines zoals Google op de hoogte zijn van de inhoud van de FAQ's, zodat zoekacties kunnen resulteren in informatie hieruit.
4 Ze willen anderen direct naar bepaalde items in de FAQ kunnen verwijzen.
5. Ze willen nieuw materiaal kunnen toevoegen aan de FAQ, maar merk op dat dit veel minder vaak gebeurt dan dat er naar antwoorden wordt gezocht. Er wordt veel meer in FAQ's gelezen dan geschreven.

Punt 1 impliceert dat de FAQ beschikbaar moet zijn in tekstformaat. De punten 2 en 3 impliceren dat de FAQ beschikbaar moet zijn als HTML-pagina, waarbij punt 2 bovendien impliceert dat de HTML moet zijn ontworpen voor leesbaarheid (d.w.z. dat u de uitstraling ervan onder controle moet houden) en een inhoudsopgave moet bevatten. Punt 4 betekent dat aan ieder afzonderlijk item in de FAQ een HTML-*'named anchor'* moet worden toegekend, een tag die mensen in staat stelt een be-

paalde plek op de pagina te bereiken. Punt 5 houdt in dat het bronbestand van de FAQ op een praktische manier beschikbaar moet zijn (zie het gedeelte 'Houd alles onder versie' in Hoofdstuk 3, *Technische infrastructuur*), in een formaat dat makkelijk kan worden aangepast.

Named Anchors en ID-attributen
Er zijn twee manieren waarop een browser naar een specifieke plek op een webpagina kan springen: named Anchors en ID-attributen.

Een *named anchor* is een normaal HTML-ankerelement (`<a>...</a>`), maar dan met een 'naam'-attribuut:

```
<a name="mylabel">...</a>
```

Recentere versies van HTML ondersteunen een generiek *id-attribuut*, dat aan ieder HTML-element kan worden toegekend, niet alleen aan `<a>`. Bijvoorbeeld:

```
<p id="mylabel">...</p>
```

Named anchors en id-attributen worden op dezelfde manier gebruikt. Iemand voegt een hekje en het label toe aan een URL zodat de browser direct naar de plek op de pagina springt:

```
http://myproject.example.com/faq.html#mylabel
```

Zo goed als alle browsers ondersteunen named anchors, de meeste moderne browsers ondersteunen ook ID-attributen. Om er zeker van te zijn dat het voor iedereen goed werkt, zou ik aanraden óf alleen named anchors te gebruiken, óf named anchors *en* ID-attributen samen (met natuurlijk hetzelfde label for beide). Named anchors sluiten zichzelf niet. Zelfs als het element geen tekst bevat, moet u het toch in de dubbelzijdige vorm schrijven:

```
<a name="mylabel"></a>
```

... hoewel er natuurlijk normaal gesproken wel tekst in staat, zoals een titel of een sectie.

Of u nu een named anchor gebruikt of een ID-attribuut of beide, houd in gedachten dat het label niet zichtbaar is voor iemand die de plek op de pagina bekijkt zonder een label te gebruiken. Deze persoon wil echter misschien de label van een bepaalde plek wel graag weten, zodat hij bijvoorbeeld de URL van een antwoord in de FAQ naar een vriend kan mailen. Om hem daarbij te helpen voegt u een *title-attribuut* toe aan hetzelfde element waar u het attribuut 'name' en/of 'id' aan hebt toegevoegd, bijvoorbeeld:

```
<a name="mylabel" title="#mylabel">...</a>
```

Wanneer de muiswijzer over de tekst gaat in een element waaraan een titel is toegewezen, laten de meeste browsers een klein boxje zien met de titel. Ik voeg meestal ook een hekje toe, om de gebruiker eraan te herinneren dat hij dit aan het eind van de URL moet zetten om de volgende keer direct naar deze plaats te springen.
Het op deze manier opmaken van de FAQ is slechts één voorbeeld van hoe u een informatiebron toonbaar kunt maken. Dezelfde eigenschappen (directe doorzoekbaarheid, beschikbaarheid voor de grote zoekmachines op internet, kunnen browsen, referentiële stabiliteit en (indien van toepassing) de mogelijkheid voor het doorvoeren van wijzigingen) zijn ook van toepassing op de andere webpagina's, het broncodeschema, de bug tracker enz. Gelukkig wordt het belang van deze eigenschappen al lang door de meeste archiveringssoftware voor mailinglijsten onderkend, zodat de meeste mailinglijsten al over deze eigenschappen beschikken. Voor andere formaten moet degene die de informatie onderhoudt misschien wat meer moeite doen (in Hoofdstuk 8, *Het managen van vrijwilligers* wordt besproken hoe u deze onderhoudstaak over verschillende vrijwilligers kunt verdelen).

Gewoontes voor codificeren
Wanneer het project groeit en complexer wordt, wordt ook de hoeveelheid informatie die iedere nieuwe deelnemer in zich op moet nemen groter. De mensen die al lange tijd bij het project betrokken zijn, konden de gewoontes van het project leren en bedenken tijdens het ontstaan ervan. Ze zijn zich er vaak niet van bewust hoe groot de hoeveelheid gewoontes na verloop van tijd is geworden en kunnen zich verbazen over hoeveel fouten nieuwkomers lijken te maken. Natuurlijk komt dit niet doordat nieuwkomers van een mindere kwaliteit zijn dan voorheen, ze moeten alleen meer moeite doen om zich te voegen in de projectcultuur dan nieuwkomers in het verleden.

De gewoontes die binnen een project worden opgebouwd, hebben minstens evenveel betrekking op hoe er moet worden gecommuniceerd en hoe informatie moet worden opgeslagen als op coderingsnormen en andere technische details. We hebben al gekeken naar de beide soorten normen, in het gedeelte 'Ontwikkelaarsdocumentatie' in Hoofdstuk 2, *Het begin* en het gedeelte 'Alles opschrijven' in Hoofdstuk 4, *Sociale en politieke infrastructuur*, waar ook voorbeelden worden gegeven. Deze sectie gaat over hoe deze richtlijnen tijdens de loop van het project up-to-date moeten worden gehouden, in het bijzonder de richtlijnen over het managen van de communicatie, omdat deze het meest veranderen wanneer de omvang en de complexiteit van een project toenemen.

Let er ten eerste op of er bij de dingen waarover mensen in de war raken patronen te herkennen zijn. Als u steeds weer dezelfde situaties ziet opduiken, met name bij nieuwe deelnemers, is de kans groot dat er een richtlijn moet worden gedocumenteerd die dat nog niet is. Laat u zich ten tweede niet ontmoedigen als u hetzelfde steeds opnieuw moet zeggen en zorg er ook voor dat u niet ontmoedigd *klinkt* doordat u een en ander steeds moet herhalen. U en de andere veteranen van het project zullen zichzelf regelmatig moeten herhalen. Dat is een onvermijdelijk neveneffect van de komst van nieuwe deelnemers.

Iedere webpagina, ieder bericht op een mailinglijst en ieder IRC-kanaal moet worden beschouwd als een reclamemogelijkheid, niet voor commerciële advertenties

maar voor aankondigingen over de eigen informatiebronnen van het project. Waarvoor u deze ruimte gebruikt hangt af van de eigenschappen van de mensen die het waarschijnlijk zullen lezen. Een IRC-kanaal voor vragen van gebruikers wordt bijvoorbeeld waarschijnlijk gebruikt door mensen die nog niet eerder met het project te maken hebben gehad. Vaak zijn dat mensen die de software net hebben geïnstalleerd en een vraag hebben waar ze graag direct antwoord op willen (want als de vraag had kunnen wachten, hadden ze die ook naar een mailinglijst kunnen sturen, wat waarschijnlijk minder tijd had gekost maar waarbij ze langer op antwoord hadden gewacht). Men is vaak niet permanent aanwezig op een IRC-kanaal. Iemand meldt zich aan, stelt zijn vraag en vertrekt weer.

Daarom moet het onderwerp via dit kanaal zijn gericht op mensen die *nu* technische antwoorden zoeken op technische vragen, in tegenstelling tot mensen die bijvoorbeeld voor langere tijd bij het project betrokken kunnen raken en voor wie richtlijnen over de omgangsvormen binnen de gemeenschap misschien handiger zijn. Dit is een voorbeeld van hoe een heel druk IRC-kanaal hiermee omgaat (vergelijk dit met het eerder genoemde voorbeeld in het gedeelte 'IRC-/ realtime-chatsystemen' in Hoofdstuk 3, *Technische infrastructuur*):

```
You are now talking on #linuxhelp

Topic for #linuxhelp is Please READ
http://www.catb.org/~esr/faqs/smart-questions.html &&
http://www.tldp.org/docs.html#howto BEFORE asking questions |
Channel rules are at http://www.nerdfest.org/lh_rules.html |
Please consult http://kerneltrap.org/node/view/799 before as-
king about upgrading to a 2.6.x kernel | memory read possible:
http://tinyurl.com/4s6mc -> update to 2.6.8.1 or 2.4.27 | hash
algo disaster: http://tinyurl.com/6w8rf | reiser4 out
```

Bij mailinglijsten is de 'advertentieruimte' een kleine voettekst die aan ieder bericht wordt toegevoegd. De meeste projecten plaatsen hier instructies over aan- en afmelden en eventueel ook een verwijzing naar de homepagina of de FAQ van het project. U denkt natuurlijk dat iedereen die bij de lijst is aangemeld deze dingen al weet, en dat is waarschijnlijk ook het geval, maar niet alleen subscribers lezen deze mailinglijstberichten. Er kunnen op vele plaatsen links te vinden zijn naar een gearchiveerd bericht en sommige berichten kunnen uiteindelijk zo algemeen bekend worden dat er meer mensen buiten de lijst zijn die het lezen dan van de lijst zelf.

Opnieuw vormgeven kan een groot verschil maken. Binnen het Subversion-project hadden we bijvoorbeeld maar weinig succes met de bugfiltertechniek zoals beschreven in het gedeelte 'Het voorfilteren van de bug tracker' in Hoofdstuk 3, *Technische infrastructuur*. Er werden nog steeds veel onterechte bugrapporten geplaatst door onervaren mensen en iedere keer dat dit gebeurde, moest degene die het rapport had ingediend op precies dezelfde manier worden geïnstrueerd als de vijfhonderd mensen vóór hem. Toen uiteindelijk één van de ontwikkelaars er niet meer tegen kon en uit zijn slof schoot tegen een arme gebruiker die de richtlijnen voor de issue tracker niet goed had gelezen, besloot een andere ontwikkelaar dat het welletjes

was. Hij stelde voor dat we de homepagina van de issue tracker opnieuw zouden vormgeven zodat het belangrijkste deel, de uitdrukkelijke opdracht een bug eerst op de mailinglijst of een IRC-kanaal te bespreken voor deze te melden, duidelijk zichtbaar zou zijn in grote rode letters op een gele achtergrond, centraal geplaatst boven alle andere elementen op de pagina. Dat deden we (het resultaat is te zien op http://subversion.tigris.org/project_issues.html) en het gevolg was een duidelijke daling van het aantal onterechte bugrapporten. Natuurlijk komen ze nog steeds binnen en dat zal altijd wel zo blijven, maar het aantal is beduidend minder, zelfs met een toenemend aantal gebruikers. Het gevolg is niet alleen dat de bugdatabase minder rommel bevat, maar ook dat de mensen die op gemelde issues reageren minder snel geïrriteerd raken en waarschijnlijk vriendelijker blijven in hun reacties op de nu minder voorkomende onterechte rapporten. Dit is een verbetering voor zowel het imago van het project als de geestelijke gezondheid van de vrijwilligers.

De les voor ons was dat het alleen opschrijven van de richtlijnen niet voldoende was. We moesten het ook ergens plaatsen waar het zichtbaar zou zijn voor de mensen die ze het meest nodig zouden hebben, en ze zo vormgeven dat de status ervan als introductieboodschap direct duidelijk is voor mensen die niet bekend zijn met het project.

Statische webpagina's zijn niet de enige plaats waar de gewoontes van het project kunnen worden geadverteerd. Een zekere mate van toezicht (in de vorm van vriendelijke geheugensteuntjes, u hoeft niet direct strenge straffen uit te delen) is ook vereist. Iedere review door collega-programmeurs, zelfs commit-reviews zoals beschreven in het gedeelte 'De code nakijken op verdachte onderdelen' in Hoofdstuk 2, *Aan de slag*, zou een onderdeel moeten bevatten waarin wordt aangegeven of wel of niet aan de normen van het project is voldaan, met name wat betreft de communicatie.

Nog een voorbeeld uit het Subversion-project. We hadden de gewoonte om 'r12908' gebruiken voor 'revisie 12908 in de versiebeheerrepository.' De klein geschreven 'r' als voorvoegsel is makkelijk te typen en omdat hij half zo hoog is als de cijfers wordt het een makkelijk te herkennen blokje tekst in combinatie met cijfers. Natuurlijk betekent het instellen van een gewoonte niet dat iedereen deze vanaf het begin consequent zal toepassen. Wanneer er dus een commit-mail binnenkomt met een logbericht als dit ...

```
------------------------------------------------------------------------
r12908 | qsimon | 2005-02-02 14:15:06 -0600 (Wed, 02 Feb 2005) |
4 lines

Patch from J. Random Contributor <jrcontrib@gmail.com>

* trunk/contrib/client-side/psvn/psvn.el:
  Fixed some typos from revision 12828.
------------------------------------------------------------------------
```

... moet de review van de commit een onderdeel bevatten met "Zou je trouwens

‘r12828’ en niet ‘revisie 12828’ willen gebruiken wanneer je refereert aan wijzigingen uit het verleden? Bedankt.” Dit is geen geleerddoenerij. Het is belangrijk, zowel voor de automatische ontleedbaarheid als de leesbaarheid.

Door het algemene principe aan te hangen dat er officieel aanvaarde referentiemethoden zijn voor gangbare stukjes informatie en dat deze referentiemethoden overal consistent moeten worden toegepast, kan het project in feite bepaalde normen exporteren. Deze normen stellen mensen in staat tools te schrijven waarmee de berichten van het project op een meer bruikbare manier worden gepresenteerd. Een revisie in de vorm ‘r12828’ kan bijvoorbeeld worden getransformeerd in een directe link naar het zoeksysteem van de repository. Dit is veel moeilijker wanneer de revisie wordt omschreven met ‘revisie 12828’, zowel omdat deze vorm kan worden opgesplitst tussen twee regels, als omdat het minder opvallend is (het woord ‘revisie’ komt vaker losstaand voor en ook groepen cijfers kunnen vaker voorkomen, terwijl de combinatie ‘r12828’ alleen een revisienummer kan zijn). Hetzelfde is van toepassing op issue-nummers, FAQ-items (hint: gebruik een URL met een named anchor, zoals beschreven in Named Anchors en ID-attributen) enz.

Zelfs voor elementen waarvoor geen duidelijke korte en algemeen aanvaarde vorm is, moeten mensen worden aangemoedigd belangrijke stukjes informatie steeds op dezelfde manier weer te geven. Als er bijvoorbeeld wordt gerefereerd aan een bericht van de mailinglijst, noem dan niet alleen de afzender en het onderwerp, maar ook de URL in het archief *en* de bericht-ID. Dit laatste geeft mensen die hun eigen kopie van de mailinglijst bewaren (mensen bewaren soms offline kopieën, bijvoorbeeld voor gebruik op een laptop onderweg) de kans het juiste bericht ondubbelzinnig terug te vinden, zelfs als ze geen toegang hebben tot de archieven. De afzender en het onderwerp zijn niet genoeg, omdat dezelfde persoon meerdere berichten kan versturen binnen dezelfde thread, zelfs op dezelfde dag.

Hoe groter een project wordt, des te belangrijker deze vorm van consistentie wordt. Consistentie houdt in dat waar mensen ook kijken, ze zien dat dezelfde patronen worden aangehouden, zodat ze deze patronen zelf ook kunnen aanhouden. Hierdoor neemt ook het aantal vragen dat ze moeten stellen af. Een miljoen lezers hebben is niet moeilijker dan één lezer. Problemen met de schaalbaarheid ontstaan pas wanneer een bepaald percentage van deze lezers vragen gaat stellen. Wanneer een project groeit, moet het dit percentage daarom verlagen door de beschikbaarheid en de toegankelijkheid van informatie te verbeteren, zodat de kans dat een willekeurig persoon vindt wat hij zoekt zonder te hoeven vragen groter wordt.

6.5 GEEN DISCUSSIES IN DE BUG TRACKER

Bij ieder project dat actief gebruik maakt van de bug tracker bestaat het gevaar dat de tracker zelf een discussieforum wordt, ook al zijn mailinglijsten daarvoor echt geschikter. Meestal begint het onschuldig. Iemand maakt een aantekening bij een issue met bijvoorbeeld een voorgestelde oplossing of een gedeeltelijk patch. Iemand anders ziet dit en realiseert zich dat de oplossing ook weer problemen met zich meebrengt. Hij voegt nog een opmerking toe, waarin hij op deze problemen wijst.

De eerste persoon reageert, weer door iets aan het issue toe te voegen enz.

Het probleem hiervan is ten eerste dat de bug tracker een nogal lompe omgeving is voor een discussie en ten tweede dat andere mensen hier misschien geen aandacht aan besteden. Uiteindelijk verwachten ze dat ontwikkelingsdiscussies op de ontwikkelingsmailinglijst worden gevoerd, en daar kijken ze dan ook. Misschien zijn ze helemaal geen subscriber van de lijst met issuewijzigingen, en zelfs als ze dat wel zijn volgen ze die misschien niet op de voet.

Maar waar ging het in het proces nou precies mis? Was dat op het moment dat de eerste persoon zijn oplossing aan het issue koppelde? Had hij deze in plaats daarvan op de mailinglijst moeten posten? Of was het de tweede persoon die reageerde bij het issue in plaats van op de lijst?

Hierop valt geen duidelijk antwoord te geven. Er is echter een algemeen principe. Als u gegevens toevoegt aan een issue kunt u dat in de tracker doen, maar als u een *discussie* begint, doe dat dan op de mailinglijst. Misschien kunt u niet altijd een duidelijk onderscheid maken, maar gebruik uw eigen beoordelingsvermogen. Als u bijvoorbeeld een patch toevoegt die een mogelijk controversiële oplossing bevat, kunt u verwachten dat mensen er vragen over hebben. Dus zelfs als u normaal gesproken een patch aan het issue zou toevoegen (ervan uitgaande dat u de wijziging niet direct kunt doorvoeren) zou u er in dit geval voor kunnen kiezen een bericht naar de mailinglijst te sturen. In ieder geval is er na verloop van tijd altijd wel iemand kan zien dat er niet meer alleen sprake is van het toevoegen van gegevens maar van een feitelijke discussie. In het voorbeeld aan het begin van deze sectie zou dat de tweede persoon zijn, die zou kunnen voorspellen dat er een discussie ontstaat op het moment dat hij zich realiseerde dat er problemen waren met de patch, en dat dit via een ander medium zou moeten gebeuren.

Om een analogie uit de wiskunde te gebruiken, wanneer informatie er uitziet alsof het snel convergeert, dan kan het in de bug tracker worden geplaatst. Wanneer het er divergerend uitziet is een mailinglijst of IRC-kanaal een betere plek.

Dit betekent niet dat er nooit enige uitwisseling van informatie kan plaatsvinden in de bug tracker. Het vragen om meer informatie over de reproductiemethode aan degene die het oorspronkelijke bericht heeft geplaatst, is bijvoorbeeld een zeer convergerend proces. Het is niet waarschijnlijk dat de reactie van die persoon nieuwe issues zal oproepen. Het zal alleen de reeds geboden informatie verder uitwerken. Het is niet nodig de mailinglijst met dit proces lastig te vallen. U kunt dit op alle mogelijke manieren afhandelen met een aantal reacties in de tracker. Hetzelfde is het geval als u er tamelijk zeker van bent dat een bug onjuist is gerapporteerd (d.w.z. dat het geen bug is). Dan kunt u dit gewoon direct binnen het issue melden. Ook het melden van een klein probleem met een voorgestelde oplossing is prima, zolang dit probleem niet de hele oplossing onderuit haalt.

Aan de andere kant, als u filosofische kwesties oprakelt over de invloed van de bug of de correcte werking van de software kunt u er redelijk zeker van zijn dat andere ontwikkelaars ook willen meepraten. Het is erg waarschijnlijk dat de discussie eerst

divergeert voordat hij weer convergeert. Daarom moet deze op de mailinglijst worden gevoerd.

Maak altijd een link vanuit het issue naar de thread in de mailinglijst als u ervoor kiest een bericht op de mailinglijst te plaatsen. Het blijft belangrijk voor iemand die het issue volgt dat hij ook de discussie kan zien, zelfs wanneer het issue zelf niet het forum is voor de discussie. Dit lijkt misschien wat omslachtig voor degene die de thread begint, maar binnen open source bestaat in principe een cultuur waarbij de schrijver verantwoordelijk is: het is veel belangrijker om dingen makkelijker te maken voor de tientallen of honderden mensen die over de bug lezen dan voor de vier of vijf mensen die erover schrijven.

Het is prima om belangrijke conclusies en samenvattingen van de discussie op de lijst naar het issue te kopiëren, als dit het voor de lezers makkelijker maakt. De meest gebruikte manier van werken is een discussie te starten op de lijst, een link naar de thread bij het issue te plaatsen en dan, wanneer de discussie afgelopen is, de uiteindelijke samenvatting in het issue te plakken (samen met een link naar het bericht waar deze samenvatting uitkomt), zodat iemand die door het issue heen bladert de conclusie snel kan zien zonder nog ergens anders naartoe te hoeven klikken. Merk op dat het gebruikelijke probleem van gegevensduplicaten hier niet bestaat omdat zowel de archieven als de commentaren bij het issue meestal statische, niet te wijzigen gegevens bevat.

6.6 PUBLICITEIT

Binnen open source-software is de grens tussen puur interne discussies en pr-verklaringen erg vaag. Dit komt voor een deel doordat de doelgroep altijd slecht gedefinieerd is: omdat de meeste of alle berichten publiekelijk toegankelijk zijn, heeft het project niet de volledige controle over de indruk die de buitenwereld ervan krijgt. Iemand, bijvoorbeeld een editor van slashdot.org, kan de aandacht van miljoenen lezers naar een bepaalde post trekken waarvan niemand ooit had verwacht dat deze buiten het project gelezen zou worden. Dit is een feit waar alle open source-projecten mee moeten leren leven, hoewel het risico in de praktijk nogal klein is. Over het algemeen zullen de aankondigingen waarvan het project het liefst wil dat ze worden gepubliceerd ook daadwerkelijk worden gepubliceerd, ervan uitgaande dat u de juiste mechanismen gebruikt om iets met nieuwswaarde aan de buitenwereld kenbaar te maken.

Voor belangrijke aankondigingen zijn er vier of vijf algemene distributiekanalen, waarop de aankondigingen zo gelijktijdig mogelijk gedaan zouden moeten worden:

1. De homepagina van uw project wordt waarschijnlijk door meer buitenstaanders gezien dan welk ander deel van het project dan ook. Wanneer u een echt belangrijke aankondiging heeft, plaats daar dan een samenvattende tekst. Deze tekst moet een zeer korte samenvatting zijn die doorlinkt naar het persbericht (zie onder) voor meer informatie.

2. Tegelijkertijd zou u op uw website een gedeelte met ‘Nieuws’ of ‘Persberichten’ moeten hebben, waar de aankondigingen gedetailleerd kunnen worden uitgewerkt. Een deel van het doel van een persbericht is een enkel, algemeen aanvaard ‘aankondigingsobject’ te bieden waar andere websites naartoe kunnen verwijzen. Zorg er dus voor dat het ook op zo’n manier is gestructureerd: óf als één webpagina per persbericht, als een afzonderlijk blogbericht of in een andere vorm waar naartoe kan worden gelinkt en waarbij het bericht afzonderlijk van andere persberichten te lezen is.

3. Wanneer uw project een RSS-feed heeft (zie het gedeelte ‘RSS-feeds’), zorg er dan voor dat de aankondiging ook daarop terecht komt. Dit kan automatisch gebeuren wanneer u het persbericht opstelt, afhankelijk van de instellingen van uw website.

4. Als uw aankondiging betrekking heeft op een nieuwe softwarerelease, wijzig dan ook de gegevens over uw project op http://freshmeat.net/ (zie het gedeelte ‘Aankondigen’ over hoe uw uw project daar moet registreren). Iedere keer dat u uw gegevens bij Freshmeat wijzigt, worden uw gegevens getoond op de wijzigingenlijst van Freshmeat voor die dag. De wijzigingenlijst wordt niet alleen op Freshmeat zelf geüpdatet, maar ook op verschillende webportalen (waaronder http://slashdot.org) die zeer goed in de gaten worden gehouden door hele hordes mensen. Freshmeat biedt dezelfde informatie ook via een RSS-feed, zodat mensen die niet zijn aangemeld bij de RSS-feed van uw project het bericht nog altijd kunnen zien via die van Freshmeat.

5. Stuur een e-mail naar de mailinglijst met aankondigingen van uw project. De naam van de lijst moet ‘announce’ zijn, dus, `announce@yourprojectdomain.org`, omdat dat inmiddels algemeen gebruik is geworden. En uit het handvest van de lijst moet duidelijk worden dat het een lijst met weinig verkeer is, gereserveerd voor belangrijke aankondigingen van het project. De meeste aankondigingen zullen betrekking hebben op nieuwe softwarereleases, maar soms ook op andere dingen, zoals een evenement voor fondsenwerving, beveiligingskwetsbaarheden (zie het gedeelte ‘Beveiligingskwetsbaarheden aankondigen’ later in dit hoofdstuk) of belangrijke wijzigingen in de leiding van het project kunnen hier worden gepost. Omdat er weinig verkeer is en alleen voor belangrijke dingen, heeft de `announce`-lijst meestal het hoogste aantal deelnemers van alle mailinglijsten van het project (dit betekent natuurlijk dat u het niet voor verkeerde doeleinden mag gebruiken. Denk dus goed na voordat u een bericht post). Om te voorkomen dat iedereen aankondigingen gaat sturen of, erger nog, er spam op de lijst terechtkomt, moet de `announce`-lijst altijd worden gemodereerd.

Probeer de aankondigingen op al deze plaatsen tegelijk te doen, of zo dicht bij elkaar als maar mogelijk is. Mensen kunnen in de war raken wanneer ze een aankondiging op de mailinglijst zien maar hier niets van terugzien op de homepagina van het project of bij de persberichten. Als u de verschillende wijzigingen (e-mails, wijzigingen van de website, etc.) in een rij plaatst en ze allemaal tegelijk verstuurt, kunt u de periode waarin de verschillende media niet consistent zijn zo kort mogelijk houden.

Voor minder belangrijke gebeurtenissen kunt u sommige of alle media weglaten. De gebeurtenis wordt door de buitenwereld toch wel opgemerkt in verhouding tot de importantie ervan. Terwijl een nieuwe softwarerelease bijvoorbeeld een belangrijke gebeurtenis is, kan alleen het noemen van een datum voor de volgende release wel enige nieuwswaarde hebben, maar is het bij lange na niet zo belangrijk als de release zelf. Wanneer een releasedatum is vastgesteld, is het de moeite waard dit feit naar de dagelijkse mailinglijst te versturen (niet naar de aankondigingenlijst) en de planning of de projectstatus op de website te updaten, maar dat is alles.

Het kan wel gebeuren dat u de datum ziet opduiken in discussies op andere plaatsen op internet, wanneer er mensen zijn die in het project geïnteresseerd zijn. Mensen die alleen maar meelezen op uw mailinglijsten en niet bijdragen, zijn niet per definitie net zo stil op andere plaatsen. Mond-tot-mondreclame zorgt voor breedschalige distributie. Daar kunt u van uitgaan en u kunt zelfs kleine aankondigingen op zo'n manier opstellen dat ze accuraat worden doorgegeven. Met name posts waarvan u verwacht dat ze zullen worden geciteerd, moeten een onderdeel bevatten dat duidelijk bedoeld is om te worden geciteerd, alsof u een formeel persbericht schrijft. Bijvoorbeeld:

hier een update van de voortgang: we zijn van plan medio augstus 2005 versie 2.0 van Scanley uit te brengen. U kunt altijd kijken op http://www.scanley.org/status.html voor updates. De belangrijkste nieuwe functie is een zoekfunctie voor letterlijke uitdrukkingen .

Andere nieuwe functies zijn: ... Er zullen ook diverse bugs worden opgelost, waaronder: ...

De eerste paragraaf is kort, geeft de twee belangrijkste stukjes informatie (de releasedatum en de belangrijkste nieuwe functie) en een URL voor aanvullende informatie. Als deze paragraaf het enige is dat iemand anders te zien krijgt, doet u het aardig goed. De rest van de e-mail kan verloren gaan zonder dat het belangrijkste deel van de inhoud hieronder te lijden heeft. Natuurlijk zullen sommige mensen een link maken naar de hele e-mail, maar evenzovaak zal slechts een klein deel van het bericht worden aangehaald. Gezien het feit dat dit laatste mogelijk is, kunt u het de ander beter makkelijk maken en uiteindelijk nog een beetje invloed houden op wat er wordt geciteerd.

Aankondigen van beveiligingskwetsbaarheden

Het omgaan met beveiligingskwetsbaarheden wijkt af van de manier waarop met andere soorten bugrapporten wordt omgegaan. Voor open source-software is open en transparant zijn een bijna heilig credo. Iedere stap van het standaardproces waarmee met bugs wordt omgegaan, is zichtbaar voor iedereen die het wil zien: de ontvangst van het eerste rapport, de daaruit volgende discussie en uiteindelijk de fix.

Bugs in de beveiliging zijn anders. Ze kunnen de gegevens van gebruikers en mogelijk hun gehele computer in gevaar brengen. Als dit probleem openlijk zou worden besproken, wordt het bestaan ervan aan de hele wereld bekendgemaakt, dus ook aan iedereen die misbruik zou willen maken van de bug. Zelfs het committen van een fix is een doeltreffende manier om anderen op de hoogte te brengen van het bestaan van de bug (er zijn potentiële aanvallers die de commit-logs van publieke projecten in de gaten houden en zoeken naar wijzigingen die wijzen op beveiligingsproblemen in de code voorafgaande aan de wijziging). De meeste open source-projecten gaan te werk met ongeveer dezelfde reeks stappen om met dit conflict tussen openheid en geheimhouding om te gaan, gebaseerd op de volgende basisrichtlijnen:

1. Praat niet publiekelijk over de bug totdat er een fix beschikbaar is. Stel de fix dan beschikbaar op exact hetzelfde moment dat u de bug bekendmaakt.

2. Kom zo snel mogelijk met de fix, met name als iemand van buiten het project de bug heeft gemeld, omdat u weet dat er minstens één persoon buiten het project is die deze kwetsbaarheid kan uitbuiten.

In de praktijk leiden deze principes tot een redelijk gestandaardiseerd stappenplan, dat in de onderstaande secties wordt behandeld.

Het rapport ontvangen

Natuurlijk moet een project in staat zijn van iedereen rapporten over beveiligingsbugs te ontvangen. Het normale adres voor het rapporteren van bugs is hier echter niet geschikt voor, omdat iedereen die kan zien. Zorg daarom voor een afzonderlijke mailinglijst voor het ontvangen van rapporten voor beveiligingsbugs. De archieven van deze mailinglijst mogen niet publiekelijk toegankelijk zijn en de deelname moet onder strenge controle staan. Alleen betrouwbare ontwikkelaars die al lang bij het project zijn betrokken, mogen deelnemen aan de lijst. Als u een meer formele omschrijving nodig hebt van 'betrouwbaar', kunt u het hebben over 'iedereen die al twee jaar of meer commit access heeft' of iets in die trant, om vriendjespolitiek te vermijden. Dit is de groep die de bugs in de beveiliging zal afhandelen.

In het ideale geval is de beveiligingslijst niet beveiligd tegen spam en niet gemodereerd, omdat u niet wilt dat een belangrijk rapport wordt uitgefilterd of vertraagd, alleen omdat geen van de moderatoren online is tijdens het weekend. Wanneer u wel automatische spambeveiligingssoftware gebruikt, probeer het dan met hoge tolerantie te configureren. Het is beter een paar spamberichten binnen te krijgen dan een rapport te missen. Om te zorgen dat de lijst werkt, moet u het adres ervan natuurlijk bekendmaken, maar gezien het feit dat deze niet gemodereerd wordt en op zijn hoogst slechts minimaal tegen spam is beveiligd, moet u proberen het adres altijd zo te transformeren dat het niet zichtbaar is, zoals beschreven in het gedeelte 'Adressen verbergen in archieven' in Hoofdstuk 3, *Technische infrastructuur*. Gelukkig hoeft het verbergen van het adres niet te betekenen dat het niet leesbaar is. Zie ook http://subversion.tigris.org/security.html en bekijk bijvoorbeeld de HTML-broncode.

De fix in stilte ontwikkelen

Wat moet de beveiligingslijst doen wanneer deze een rapport ontvangt? Het eerste dat er gedaan moet worden, is het evalueren van de ernst en de urgentie van het probleem:

1. Hoe ernstig is de kwetsbaarheid? Geeft het een kwaadwillende aanvaller gelegenheid de computer van iemand anders die de software gebruikt over te nemen? Of is het probleem alleen dat er informatie uitlekt over de omvang van enkele van hun bestanden?

2. Hoe makkelijk kan de kwetsbaarheid worden uitgebuit? Kan een aanval worden omgezet in een script of is er kennis nodig over de omstandigheden, gefundeerde gissingen en puur geluk?

3. *Wie* heeft het probleem gerapporteerd? Het antwoord op deze vraag verandert natuurlijk niets aan de aard van de kwetsbaarheid, maar het geeft u een idee over hoeveel mensen er van af kunnen weten. Als het rapport komt van één van de eigen ontwikkelaars van het project kunt u een beetje opgelucht ademhalen (maar alleen een klein beetje), omdat u er op kunt vertrouwen dat deze niemand anders over het probleem heeft verteld. Aan de andere kant, als de e-mail van `anonymous14@globalhackerz.net` kwam, dan kunt u beter zo snel mogelijk reageren. Deze persoon heeft u een gunst bewezen door u erover te informeren, maar u hebt geen idee hoeveel mensen hij erover heeft verteld of hoelang hij zal wachten voordat hij de kwetsbaarheid gaat uitbuiten op in gebruik zijnde systemen.

Merk op dat het verschil waar we het hier over hebben vaak een heel dun lijntje is tussen *urgent* en *extreem urgent*. Zelfs als het rapport van een bekende en goedwillende bron afkomstig is, dan nog kunnen er anderen zijn op internet die de bug lang geleden al hebben ontdekt en dit alleen niet hebben gerapporteerd. Het probleem is alleen niet urgent als de bug de beveiliging niet ernstig in gevaar brengt.

Het voorbeeld "`anonymous14@globalhackerz.net`" is trouwens niet grappig bedoeld. U kunt werkelijk bugrapporten krijgen van mensen die hun identiteit geheim houden en van wie u uit hun woorden en gedrag niet goed kunt opmaken of ze wel of niet aan uw kant staan. Het maakt in feite niet uit: als ze het gat in de beveiliging aan u rapporteren, zullen ze het gevoel hebben dat ze een goede daad hebben gedaan en moet u dienovereenkomstig reageren. Bedank de persoon voor het rapport, geef hem een datum waarop u uiterlijk van plan bent een fix te publiceren en houd hem op de hoogte. Soms geeft de rapporteur *u* een datum. Dat is dus een impliciet dreigement om de bug op een bepaalde datum te publiceren, of u er nu klaar voor bent of niet. Dit kan overkomen als een machtsspelletje van een kwelgeest, maar de kans is groot dat het een preventieve actie is, volgend op een teleurstelling over niet reagerende softwareproducenten die een beveiligingsrapport niet serieus genoeg namen. Maar hoe het ook zit, u kunt het zich niet veroorloven deze persoon een veeg uit de pan te geven. Als de bug ernstig is, heeft hij uiteindelijk de kennis in huis om bij uw gebruikers grote problemen te veroorzaken. Behandel dergelijke rapporteurs goed en hoop dat ze u ook goed zullen behandelen.

Een andere veel voorkomende rapporteur over beveiligingsbugs is de beveiligingsprofessional, iemand die zijn geld verdient met het controleren van code en op de hoogte is van de laatste nieuwtjes over softwarekwetsbaarheden. Deze mensen hebben vaak ervaring met beide kanten van het verhaal. Ze hebben zowel rapporten verzonden als ontvangen, en waarschijnlijk vaker dan de meeste ontwikkelaars in uw project. Ook zij geven vaak een deadline voor het oplossen van de kwetsbaarheid voordat ze de bug bekend zullen maken. De deadline kan onderhandelbaar zijn, maar dat is aan de rapporteur. Beveiligingsprofessionals zijn het er over eens dat deadlines ongeveer de enige betrouwbare manier zijn om ervoor te zorgen dat organisaties direct reageren op beveiligingsproblemen. Behandel de deadline dus niet als onbeleefd. Het gaat om een reeds lang bestaande traditie waar een goede redenen voor zijn.

Zodra u weet hoe ernstig en urgent het probleem is, kunt u aan de oplossing gaan werken. Soms moet een compromis worden gesloten tussen een elegante en een snelle oplossing. Daarom moet u het eens worden over de urgentie voordat u van start gaat. Beperk de discussies over de fix uiteraard tot de leden van de beveiligingslijst en de oorspronkelijke rapporteur (als hij op de hoogte gehouden wil worden) en andere ontwikkelaars die er om technische redenen bij betrokken moeten worden.

Voeg de fix niet toe aan de repository. Houd de patch-vorm aan tot het moment van publicatie Als u een commit invoert kan iemand zelfs uit een onschuldig ogend logbericht begrijpen waar de verandering over gaat. U weet maar nooit wie er meekijkt in uw database en wie er geïnteresseerd kan zijn. Het uitzetten van de commit-e-mails helpt ook niet. Ten eerste is een hiaat in de reeks commit-mails op zich al verdacht, en bovendien blijven de gegevens toch nog beschikbaar in de repository. Ontwikkel de hele fix in de vorm van een patch en bewaar de patch op een veilige plek, misschien in een afzonderlijke en persoonlijke database die alleen bekend is bij mensen die al op de hoogte waren van de bug. (Als u een gedecentraliseerd versiebeheersysteem zoals Arch of SVK gebruikt, kunt u het werk doen onder volledige versiebeheer en hoeft u de database alleen maar ontoegankelijk te houden voor buitenstaanders.)

CAN/CVE-nummers

Mogelijk hebt u een *CAN-nummer* of een *CVE-nummer* gezien dat geassocieerd wordt met beveiligingsproblemen. Deze nummers kunnen er bijvoorbeeld zo uit zien: 'CAN-2004-0397" of "CVE-2002-0092'.

Beide soorten nummers staan voor dezelfde soort entiteit: een post op de lijst met 'Common Vulnaribilities and Exposures (algemene kwetsbaarheden en blootstellingen)' die wordt bijgehouden op http://cve.mitre.org/. De bedoeling van de lijst is het bieden van gestandaardiseerde namen voor alle bekende beveiligingsproblemen, zodat iedereen een unieke en algemeen aanvaarde term kan gebruiken bij de bespreking ervan, en een centrale plek waar men terecht kan voor informatie. Het enige verschil tussen een 'CAN'-nummer en een 'CVE'-nummer is dat de eerste staat voor een mogelijk entry die nog niet is goedgekeurd voor opname op de officiële lijst bij het CVE-bestuur. Het tweede nummer staat voor een goedgekeurde entry. Beide soorten entry's zijn echter zichtbaar voor het publiek, en een entry-nummer verandert niet als het wordt goedgekeurd. In dat geval wordt het voorvoegsel 'CAN' gewoon vervangen door 'CVE'.

Een CAN/CVE-entry bevat zelf geen volledige beschrijving van de bug en hoe u zichzelf ertegen kunt beschermen. In plaats daarvan geeft het een korte samenvatting en een lijst met referenties naar externe informatiebronnen (zoals mailinglijstarchieven) waar mensen kunnen kijken voor meer gedetailleerde informatie. Het werkelijk doel van http://cve.mitre.org/ is een goed georganiseerde plek te bieden waar iedere kwetsbaarheid een naam kan krijgen en een duidelijke aanwijzing naar meer informatie. Zie http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002-0092 voor een voorbeeld van een entry. Merk op dat de referenties heel erg beknopt kunnen zijn en de informatiebronnen vermeld zijn in de vorm van cryptische afkortingen. Een verklaring van de afkortingen kan worden gevonden op http://cve.mitre.org/data/refs/refkey.html.

Als uw kwetsbaarheid voldoet aan de CVE-criteria kunt u een CAN-nummer krijgen. De procedure daarvoor is met opzet moeilijk gemaakt. Het komt er in feite op neer dat u iemand moet kennen, of iemand moet kennen die weer iemand kent. Dat is niet zo gek als het misschien lijkt. Om te voorkomen dat het CVE-bestuur wordt overladen met onechte of slecht geschreven voorstellen, accepteren ze alleen voorstellen van mensen die ze kennen en vertrouwen. Om er dus voor te zorgen dat uw kwetsbaarheid wordt opgenomen, moet u dus een lijntje van kennissen zien te vinden tussen uw project en het CVE-bestuur. Doe eens navraag bij uw ontwikkelaars. Eén van hen kent vast wel iemand die óf zelf een CAN-proces heeft doorlopen óf die iemand kent die dat heeft gedaan enz. Het grote voordeel hiervan is ook dat ergens in de keten iemand kan zitten die genoeg van de materie afweet om u te vertellen dat a) het probleem niet wordt aangemerkt als kwetsbaarheid of blootstelling volgens de MITRE-criteria en het dus geen zin heeft een voorstel in te dienen of b) de kwetsbaarheid al een CAN- of CVE-nummer heeft. Dit laatste kan het geval zijn wanneer een bug al eerder is gepubliceerd op een andere beveiligingslijst, bijvoorbeeld op http://www.cert.org/ of op de BugTraq-mailinglijst op http://www.securityfocus.com/. (Als dit is gebeurd zonder dat uw project hierover gehoord heeft, moet u zich wel zorgen gaan maken over wat er allemaal nog meer aan de hand kan zijn waar u niets vanaf weet.)

Als u wel een CAN/CVE-nummer kunt krijgen, wilt u dat normaal gesproken zo vroeg mogelijk in het proces van onderzoek naar de bug hebben, zodat alle toekomstige communicatie naar dat nummer kan verwijzen. CAN-entry's zijn geblokkeerd tot de datum van publicatie. De entry is zichtbaar als lege tijdelijke aanduiding (zodat u de naam niet kwijtraakt), maar laat geen informatie zien over de kwetsbaarheid tot het moment dat u de bug en de fix bekendmaakt.

Meer informatie over de CAN/CVE-procedure kunt u vinden op http://cve.mitre.org/about/candidates.html. Een bijzonder heldere uiteenzetting over het gebruik van CAN/CVE-nummers in een project kunt u vinden op http://www.debian.org/security/cve-compatibility.

Vooraankondiging

Zodra uw beveiligingsteam (dat zijn de ontwikkelaars op de beveiligingslijst of degenen die u bij het team hebt gehaald om een bepaald probleem op te lossen) een fix klaar heeft, moet u beslissen hoe u deze verspreidt.

Als u de fix gewoon aan uw repository toevoegt of deze op een andere manier aan de hele wereld bekendmaakt, dwingt u iedereen die uw software gebruikt om onmiddellijk te upgraden, anders lopen ze het gevaar te worden gehackt. Daarom kan het soms wenselijk zijn bij bepaalde belangrijke gebruikers een *vooraankondiging* te doen. Dit is met name van belang voor client/serversoftware, in het geval van bekende servers die een aantrekkelijk slachtoffer kunnen zijn voor aanvallers. De beheerders van deze servers hebben graag een dag of twee extra voor hun upgrade, zodat ze al beschermd zijn op het moment dat het probleem wereldkundig wordt gemaakt.

Een vooraankondiging betekent dat er e-mails worden verzonden naar deze beheerders voorafgaande aan de bekendmakingsdatum, waarin hun wordt verteld over de kwetsbaarheid en de oplossing daarvoor. U zou alleen een vooraankondiging moeten sturen naar mensen bij wie u er op kunt vertrouwen dat zij discreet met de informatie om zullen gaan. Dat wil zeggen dat de ontvanger van de vooraankondiging aan twee voorwaarden moet voldoen: de ontvanger moet een grote, belangrijke server beheren waarvoor een bedreiging een serieus probleem zou zijn *en* de ontvanger moet erom bekendstaan dat hij zijn mond vóór de bekendmakingsdatum niet voorbijpraat over het beveiligingsprobleem.

Stuur iedere vooraankondiging afzonderlijk (één per keer) naar iedere ontvanger. Stuur het bericht *niet* naar alle ontvangers in één keer, omdat ze elkaars namen dan kunnen zien, wat betekent dat u in principe alle ontvangers vertelt dat iedere *andere* ontvanger een gat in de beveiliging van zijn server zou kunnen hebben. Versturen via een blinde bcc is ook geen goed idee, omdat sommige beheerders hun inboxen beschermen met spamfilters die met bcc verstuurde e-mails blokkeren of een lagere prioriteit toekennen, omdat er tegenwoordig veel spam met bcc wordt verstuurd.

Hier is een eenvoudig voorbeeld van een vooraankondigingsmail:

```
Van: Uw naam
Aan: admin@large-famous-server.com
Reply-to: Uw naam (niet het adres van de beveiligingslijst) Onderwerp: Vertrouwelijke aankondiging kwetsbaarheid Scanley.

Deze e-mail is een vertrouwelijke vooraankondiging van een beveiligingswaarschuwing in de Scanley-server.

Stuur dit bericht of delen ervan *niet door* naar anderen. De publieke bekendmaking is pas op 19 mei en we willen de informatie graag tot die datum geheim houden.

Wij sturen u deze e-mail omdat u (volgens onze informatie) een Scanley-server heeft en dat u de patch zou willen installeren voordat het gat in de beveiliging op 19 mei wordt bekendgemaakt.
```

```
Referenties:
============
CAN-2004-1771: Scanley stack overflow in queries

Kwetsbaarheid:
==============
Er kunnen willekeurige commando's vanaf de server worden gestart
wanneer de locale van de server niet correct is geconfigureerd en
de client een misvormde query verstuurt.

Ernst:
======
Zeer ernstig, er kan willekeurige code worden uitgevoerd op de
server.

Tijdelijke oplossing:
====================
Stel de instelling voor 'natural-language-processing' op 'off'
in scanley.conf, deze kwetsbaarheid wordt hiermee opgeheven.

Patch:
======
De onderstaande patch heeft betrekking op Scanley 3.0, 3.1 en
3.2.

Er komt een nieuwe publieke release uit (Scanley 3.2.1) op of
kort voor 19 mei, zodat deze beschikbaar is op hetzelfde moment
dat deze kwetsbaarheid bekend wordt gemaakt.  U kunt de patch nu
uitvoeren of wachten op de publieke release.  Het enige verschil
tussen 3.2 en 3.2.1 is deze patch.

[...voeg de patch hier in...]
```

Als u een CAN-nummer heeft, voeg dit dan aan de vooraankondiging toe (zoals hierboven getoond), zelfs als de informatie nog steeds geblokkeerd is en de MITRE-pagina niet zichtbaar zal zijn. Door het CAN-nummer in te voegen weet de ontvanger zeker dat de bug waarover de vooraankondiging heeft ontvangen dezelfde is als de bug waarover hij later via de publieke kanalen te horen krijgt, zodat hij zich geen zorgen hoeft maken of hij nog meer actie moet ondernemen, wat precies de bedoeling is van CAN/CVE-nummers.

De fix publiceren

De laatste stap in het oplossen van een beveiligingsbug is het distribueren van de fix. U zou het probleem in één uitgebreide aankondiging moeten beschrijven, eventueel het CAN/CVE-nummer geven, aangeven hoe de bug tijdelijk kan worden omzeild en hoe deze permanent moet worden opgelost. Meestal betekent 'oplossen' het upgraden naar een nieuwe versie van de software, hoewel het soms ook het uitvoeren van een patch kan inhouden, met name wanneer de software normaal gesproken al in source-vorm loopt. Als u wel een nieuwe release uitbrengt, moet deze alleen met deze patch afwijken van de vorige release. Daardoor kunnen meer behoudende beheerders upgraden zonder zich zorgen te hoeven maken over wat er nog meer verandert. Ze hoeven zich ook niet druk te maken over toekomstige upgrades, omdat de beveiligingspatch per definitie in alle toekomstige releases is opgenomen. (De details van releaseprocedures worden verder besproken in het gedeelte 'Beveiligingsreleases' in Hoofdstuk 7, *Het maken van downloadpakketten, releases en dagelijkse ontwikkeling*.)

Of u de publieke fix nu wel of niet uitbrengt in een nieuwe release, doe de aankondiging met ongeveer dezelfde prioriteit als waarmee u een nieuwe release zou uitbrengen: stuur een e-mail naar de `announce`-lijst van het project, maak een nieuw persbericht, update de entry bij Freshmeat enz. Hoewel u het bestaan van een beveiligingsbug nooit mag marginaliseren om de reputatie van het project niet te schaden, kunt u de toon en het belang van een beveiligingsaankondiging wel zo treffen dat deze overeenkomt met de feitelijke ernst van het probleem. Als het gat in de beveiliging slechts een minimaal probleem is voor het lekken van informatie en geen groot probleem waarbij de hele computer van de gebruiker kan worden overgenomen, dan hoeft er ook niet veel ophef over te worden gemaakt. U kunt zelfs beslissen de `announce`-lijst hier helemaal niet mee lastig te vallen. Als het project namelijk steeds vals alarm slaat, kunnen gebruikers uiteindelijk gaan denken dat de software minder veilig is dan werkelijk het geval is en zullen ze u misschien ook niet meer geloven wanneer er wel een groot probleem wordt aangekondigd. Zie http://cve.mitre.org/about/terminology.html voor een goede inleiding over het inschatten van de ernst van een probleem.

Als u niet weet hoe u met een beveiligingsprobleem om moet gaan, kunt u over het algemeen het beste met iemand praten die er ervaring mee heeft. Het beoordelen en oplossen van kwetsbaarheden is in grote lijnen een vaardigheid die u kunt aanleren en u kunt de eerste paar keer makkelijk fouten maken.

22| Er is interessant wetenschappelijk onderzoek gedaan over dit onderwerp. Zie bijvoorbeeld *Group Awareness in Distributed Software Development* van Gutwin, Penner en Schneider. Dit rapport was een tijdje online beschikbaar, vervolgens een poosje niet, en nu weer wel op http://www.st.cs.uni-sb.de/edu/empirical-se/2006/PDFs/gutwin04.pdf. U kunt deze link dus proberen, maar misschien moet u een zoekmachine gebruiken als de locatie weer veranderd is.