UNIX System Security

Recente Ervaringen en Aanbevelingen

Edwin H. Kremer

RUU-CS-91-33
September 1991

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

UNIX System Security

Recente Ervaringen en Aanbevelingen

Edwin H. Kremer

Technical Report RUU-CS-91-33
September 1991

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924–3275

UNIX System Security
Recente Ervaringen en Aanbevelingen

Edwin H. Kremer
Vakgroep Informatica
Rijksuniversiteit Utrecht
(edwin@cs.ruu.nl)

Samenvatting

Dit artikel gaat in op de security problemen van UNIX systemen. Het beperkt zich voornamelijk tot de meest triviale en alledaagse zaken die juist daarom veelal over het hoofd gezien worden. Het doel van dit artikel is gebruikers en systeembeheerders te helpen bij het verhogen van de security van het systeem waarmee zij werken cq. waarvoor zij verantwoordelijk zijn. Problemen worden gesignaleerd en oplossingen worden aangereikt. Uit dit artikel zal blijken dat de geboden oplossingen simpel en toch bijzonder doeltreffend zijn. Hoewel enkele hoofdstukken een behoorlijke voorkennis van UNIX vereisen en voornamelijk voor systeembeheerders geschreven zijn, is het overgrote deel van dit artikel ook zeer goed leesbaar voor hen die niet over deze voorkennis beschikken.

Inhoudsopgave

1 Voorwoord

Ik, een computer–kraker? Nee hoor, enkel een UNIX systeembeheerder met een waarschijnlijk iets meer dan gemiddelde interesse voor UNIX system security. Aangespoord door een aantal recente slechte ervaringen met de beveiliging van UNIX systemen, ben ik maar eens in de huid van een cracker[1] gekropen om zelf uit te vinden hoe het nu werkelijk met die security gesteld is.

Doel was vooral om na te gaan wat leveranciers en systeembeheerders geleerd hebben van de **Internet worm** [Denn89, Donn89, Eich89, Eise89, Seel89, Spaf88, Spaf89] die in November 1988 opdook in het Amerikaanse DARPA Internet. Daarnaast heb ik gepoogd inzicht te krijgen in de wijze waarop gebruikers omgaan met *passwords* en wat verantwoordelijke instanties (software leveranciers en systeembeheerders) ondernemen om een betere password beveiliging te verkrijgen. Eigenlijk komt het erop neer dat ik op een meer gerichte manier heb geprobeerd een moderne variant van de Internet worm na te spelen. Mijn bevindingen zijn in dit artikel verwerkt.

De aanleiding tot dit onderzoek was een aantal gevallen waarin de beveiliging van een UNIX systeem door crackers op de proef werd gesteld en een aantal gevallen waarin door crackers daadwerkelijk de beveiliging doorbroken werd en behoorlijke schade aangericht is. Een paar recente kraken en pogingen daartoe binnen Nederland zal ik eveneens toelichten.

Een probleem bij het schrijven over security is het vinden van een balans tussen wat men vertelt en wat men verzwijgt: vertelt men te veel, dan worden kwade geesten wellicht op slechte ideeën gebracht; vertelt men te weinig, dan wordt het probleem niet als probleem gezien en is alle moeite voor niets geweest. In dit artikel heb ik geprobeerd een middenweg tussen deze twee uitersten te vinden. Ik, een cracker? Nee hoor, absoluut niet!

[1] Een **cracker** is iemand die probeert de beveiliging van een computersysteem te breken. Hij moet vooral niet verward worden met het wizard-type dat **hacker** genoemd wordt.

2 Inleiding

Dit inleidende hoofdstuk geeft een kort overzicht van de methoden die een cracker volgt om in een systeem in te breken en een korte toelichting op de terminologie die in de rest van dit artikel gebruikt wordt.

In hoofdstuk 3 worden lekken in het UNIX operating system beschreven. Hoofdstuk 4 gaat in op het password mechanisme in UNIX en legt de zwakheden ervan bloot. Hoofdstuk 5 maakt duidelijk wat systeembeheerders kunnen doen om de security van hun systeem te verbeteren. Hoofdstuk 6 beschrijft wat "normale" gebruikers kunnen doen om de security te verbeteren en geeft tevens adviezen voor een goed gebruik van passwords; dit hoofdstuk is zeker ook voor systeembeheerders van belang. Hoofdstuk 7 tenslotte, gaat in op een aantal recente incidenten, zowel inbraakpogingen als geslaagde kraken.

2.1 Hoe vinden inbraken plaats?

In het algemeen verkrijgen crackers ongewenst toegang tot een computersysteem door ofwel een lek[2] in de beveiliging ofwel het raden van het password van een legitieme gebruiker. Als een cracker via een lek in de beveiliging toegang tot een systeem verkrijgt, blijkt achteraf vaak dat het bestaan van dit lek al lang bekend was: blijkbaar heeft de leverancier of de systeembeheerder het lek nooit belangrijk genoeg gevonden om er wat aan te doen.

Passwords die gekraakt worden, blijken veelal passwords te zijn die een zekere herkenbare relatie met de eigenaar hebben (achternaam, auto-kenteken, naam van partner, etc.). Blijkbaar is de gebruiker wiens password geraden is, zich niet bewust geweest van het belang van het kiezen van een moeilijk te achterhalen password. Men zou ook kunnen stellen dat het systeem niet voldoende eisen stelt aan het te gebruiken password.

2.2 Toelichting op de gebruikte terminologie

In dit artikel worden nogal wat "standaard UNIX" termen gebruikt, die voor een deel van het lezerspubliek misschien enige toelichting behoeven.

In UNIX bestaan twee soorten objecten: *files* en *processen*. Beide objecten (resources) hebben een binding met een gebruiker: een gebruiker is *eigenaar* van een file, en een door de gebruiker gestart proces heeft dezelfde *permissies* die de gebruiker heeft.

Het UNIX operating system wijst aan iedere gebruiker **twee** identificatie sleutels toe: de **uid** (= user identification) en de **gid** (= group identification). Deze twee sleutels bepalen samen wat een gebruiker wel en niet mag en verdelen de gebruikers op een systeem in drie categoriën: zij die over de *uid* sleutel beschikken, zij die over de *gid* sleutel beschikken en zij die over geen van beide sleutels beschikken. Deze drie categoriën komen precies overeen met de drie beveiligingsniveaus op *file* niveau: **owner**ship, **group** membership en **others**. Een gebruiker is dus òf **owner** van een file (uid komt overeen met die van de file), òf is lid van de **group** van die file (gid komt overeen met die van de file) òf is een volkomen vreemde voor die file.

[2] In het dagelijks spraakgebruik heet dit meestal een *security hole* of een *security flaw*.

De recente UNIX systemen gaan een stap verder door aan iedere file een *access-control-list* te hangen, die het mogelijk maakt om per file te specificeren welke gebruikers wat met de betreffende file mogen doen, bijvoorbeeld [user A: lezen], [user B: lezen, schrijven], etc.

Processen in UNIX hebben, naast de uid en de gid, ook nog twee andere identificatie sleutels: de *euid* (effective uid) en de *egid* (effective gid). In het merendeel van de gevallen zijn de euid en de egid identiek aan respectievelijk de uid en de gid. Het nut van deze extra id's schuilt in een zeer krachtig mechanisme in UNIX: *set-uid-on-execution* en *set-gid-on-execution*. Dit mechanisme maakt het mogelijk om tijdelijk de effectieve id van iemand anders aan te nemen, met de mogelijkheid de eigen identiteit (uid/gid) in een later stadium weer op te roepen. Een programma dat "set-uid gebruiker A" is, zal altijd draaien met alle rechten die gebruiker A heeft, ook als het programma door gebruiker B aangeroepen wordt. Een toepassing hiervan volgt hieronder.

Er is één speciale gebruiker in UNIX: de **super-user**. Dit is de gebruiker wiens loginnaam meestal *root* is. De super-user (veelal de systeembeheerder) is de gebruiker die *alles* met het systeem mag doen: op hem zijn de genoemde protectie-mechanismen niet van toepassing. Een willekeurige gebruiker die de euid van de super-user weet aan te nemen, mag dus ook alles met het systeem doen. Een mooie toepassing van het set-uid-on-execution mechanisme, vindt men in het commando df dat laat zien wat de bezettingsgraad van de disks is. Om achter de bezettingsgraad van een disk te komen, is het noodzakelijk de administratieve data waarin de disk-usage gegevens staan, van die disk te lezen. Als iedereen dit zou kunnen, dan zou iedereen dus ook **alle** data van die disk kunnen lezen, daarmee de normale protecties van het filesysteem omzeilend. De enige gebruiker die wel de disk-usage van de disk mag lezen, is de super-user. Men wil andere gebruikers dus alleen *gecontroleerd* dezelfde rechten geven die de super-user heeft, m.a.w. iedere gebruiker mag super-user rechten hebben zolang hij bezig is met het lezen van de disk-usage data, dus voor de levensduur van het df commando, maar niet langer. Dit kan bereikt worden met het *set-uid-on-execution* mechanisme. Men zegt dan "df is set-uid root".

Tot dusver hebben we ons alleen beziggehouden met mechanismen voor de bescherming van gebruikersdata tegen andere gebruikers op hetzelfde systeem. **Networking**[3] voegt een nieuwe dimensie toe: voor systemen (hosts) onderling geldt min of meer wat voor gebruikers onderling geldt. De ene host moet data afschermen voor gebruik vanaf een andere host, maar tegelijkertijd moeten hosts onderling eenvoudig en gecontroleerd data kunnen delen. Ieder systeem krijgt een eigen identificatie sleutel toegewezen, de **hostid** of **host-name**; vergelijkbaar met de uid van een gebruiker. Als bijvoorbeeld de systeembeheerder van host X vindt dat hosts Y en Z toegang moeten hebben tot de data van host X, dan worden hosts Y en Z *trusted hosts* van host X genoemd. Dit heeft ook consequenties voor de gebruikers: een gebruiker op host Y of Z heeft dan ook direct toegang tot host X zonder dat hij zich eerst nog moet identificeren: de identificatie heeft immers al op host Y of Z plaatsgevonden.

Belangrijk is nog te weten dat de systeem-software geen onderscheid maakt tussen *lokale* en *niet-lokale*[4] hosts; er is dus geen sprake van een algemeen "ingebouwd" basis beveiligings-niveau gebaseerd op de logische plaats van een host in een netwerk.

[3] *Networking* mag gelezen worden als een algemene term voor de complexe wereld van computers die via netwerken verbonden zijn, gedistribueerde filesystemen, shared databases, etc.

[4] Niet-lokale hosts zijn systemen die onder de verantwoording van een andere systeembeheerder vallen.

3 Lekken in de UNIX Operating System beveiliging

UNIX is vanaf het begin een *open systeem* geweest: het was vooral belangrijk dat alle resources voor iedereen toegankelijk waren. Vanuit dit oogpunt bezien, zou je kunnen zeggen dat het met de beveiliging van UNIX al bij de geboorte fout zat, maar dat is niet helemaal waar. Het basis *file protectie* mechanisme van UNIX biedt de mogelijkheid een redelijk veilig systeem te maken, maar naarmate UNIX groeide en *networking* toegevoegd werd, werd duidelijk dat er meer en betere security mechanismen moesten komen. De vraag hiernaar nam enorm toe, nadat UNIX ook gebruikt ging worden bij instanties waar bescherming van data zeer belangrijk is. In de recente UNIX versies is het file protectie mechanisme uitgebreid met de in sectie 2.2 genoemde *access-control-lists.*

Naast een uitbreiding van het protectie mechanisme kwam er ook meer aandacht voor de kwaliteit van de systeem-software. Ontwerpfouten en achterdeurtjes die men heeft vergeten dicht te timmeren, leiden ook vandaag de dag nog tot situaties waarin crackers relatief eenvoudig toegang tot een systeem kunnen verkrijgen. In de volgende secties zal ik kort de meest beruchte lekken in de systeem-software bespreken; een paar van deze lekken kunnen ook vandaag de dag nog in een systeem aanwezig zijn.

3.1 Het recover mechanisme van de VI editor

Dit is een geval uit de oude doos, maar een prachtig voorbeeld waarin de software-ontwerper net iets te veel aandacht besteed heeft aan het gemak voor de gebruiker: hij ging er vanuit dat "alles runtime door de gebruiker te configureren moet zijn". Een zeer te waarderen uitgangspunt dat in dit geval niet geldig blijkt te zijn.

De **vi** editor wordt binnen UNIX ondersteund door een *recovery systeem* dat er voor moet zorgen, dat de gebruiker bij een plotselinge system-crash of een ander abnormaal einde van zijn editor-sessie niet voor niets heeft zitten tikken. Zodra het systeem weer functioneert, moet de gebruiker zijn werk weer eenvoudig kunnen oppakken. Zoals te begrijpen is, dient het recovery mechanisme onvoorwaardelijk toegang te hebben tot alle systeem resources. Immers, zelfs als de gebruiker een file aan het editen was die alleen voor hem toegankelijk was, dan nog moet het recovery systeem deze file kunnen benaderen om in het geval van een system-crash de reddingsoperatie correct uit te kunnen voeren.

Tot dusver is er geen vuiltje aan de lucht, maar de ontwerper bedacht dat het ook wel netjes zou zijn, als de gebruiker via e-mail op de hoogte gebracht zou worden van het feit dat zijn file netjes ge-recovered is door het systeem. Nu zijn er vele mail-programma's (mailers), dus waarom de gebruiker niet zijn favoriete mailer laten kiezen en deze gebruiken om de mail te sturen? Hiertoe keek het recovery systeem naar de environment variabele **$MAILER**. Het resultaat moge duidelijk zijn: men kon met super-user permissie een zelf-gedefiniëerd programma starten, waarin men zich bijvoorbeeld voor langere tijd super-user permissie kon toeëigenen. Bijvoorbeeld het shell-script *badguy.sh*:

```
cp /bin/sh /tmp/rootsh ; chmod u+s,go=rx /tmp/rootsh
```

Het enige wat nu nog rest is dit shell-script toe te kennen aan de environment variabele **$MAILER**,

```
$ export MAILER=$HOME/badguy.sh
```

een editor sessie starten en deze vervolgens op een ongebruikelijke wijze om zeep helpen...

3.2 At – executeer commando's op een later tijdstip

Ook dit is een geval uit de oude doos, maar de mogelijkheid bestaat dat deze fout ook nu nog in vele systemen aanwezig is. Vandaar dat ik niet te ver in details zal treden. Security holes in *at* komen voor in de oudere Ultrix systemen (Versie 2.x) en in de oudere AT&T System V releases. Op beide genoemde systemen kijkt de daemon[5], die de "at" jobs uiteindelijk executeert, naar de *owner* van de file om te bepalen met wiens rechten de job moet draaien. Op beide systemen bleek het echter bijzonder eenvoudig te zijn de *submitted job* alsnog van eigenaar te doen veranderen.

In Ultrix was het mogelijk de job door *root* uit te laten voeren, vanwege verkeerde permissies (`drwxrwxrwx`) op de spool directory `/usr/spool/at`. In System V was het mogelijk de job aan een ander te geven, omdat System V toestaat dat men zijn owner-rechten op een file weggeeft aan een andere gebruiker.

3.3 Sendmail – de mailer daemon

Misschien wel de meest beruchte security hole was de *debug* faciliteit van de **sendmail** daemon, de min of meer defacto standaard UNIX mail transfer agent op 4.3BSD en aanverwante systemen. Sendmail is één van de vele daemons in UNIX die met super-user permissie moeten draaien om correct te kunnen werken. Tijdens het ontwikkelen van sendmail bedacht de ontwerper dat het handig zou zijn, als de daemon op afstand in een *debug-mode* gezet zou kunnen worden: dit vereenvoudigde het debuggen in de test fase. In deze debug-mode staat sendmail ook toe dat er mail gestuurd wordt aan een **programma** in plaats van aan een gebruiker. Uitgaande van het feit dat de daemon met super-user permissie draait en dat een **programma** en niet een mailbox de mail ontvangt, moge het duidelijk zijn dat dit interessante mogelijkheden voor misbruik biedt, vooral als het betreffende programma de standaard command interpreter `/bin/sh` is.

Het was o.a. deze security hole die met veel succes door de ontwerper van de *Internet worm* gebruikt werd. Nu, bijna drie jaar na de Internet worm, blijkt nog steeds een behoorlijk aantal systemen langs deze weg kwetsbaar te zijn. En dan te bedenken dat er reeds tijdens het gevecht tegen de *worm* patches[6] voor sendmail beschikbaar kwamen die afdoende werkten...

3.4 FTP – File Transfer Protocol

Verspreiding van software vindt binnen de UNIX community veelal plaats door middel van een zgn. *anonymous FTP* service. Over de gehele wereld zijn honderden sites. die via een *FTP-site* gecontroleerd software en documentatie beschikbaar stellen aan derden[7]. Omdat het hier om een *anonieme* service gaat, dus een service waarbij men zijn identiteit niet behoeft te onthullen, is het niet de bedoeling dat men via deze service meteen toegang krijgt tot het gehele systeem waarop men gast is.

[5] Een daemon is een altijd aanwezig process dat het merendeel van de tijd niets anders doet dan wachten op werk.

[6] Een patch (= acroniem voor "Please apply this clever hack") is een modificatie op bestaande software en wordt veelal verstrekt als het niet nodig is een volledige nieuwe release van de betreffende software te verspreiden.

[7] Zie voor een voorbeeld appendix A.

Een correct geconfigureerde FTP service plaatst de gebruiker dan ook in een afgeschermd, klein stukje van het UNIX filesysteem. Tevens krijgt de gebruiker, naast ingebouwde FTP commando's als *get* en *put*, slechts de beschikking over een enkel UNIX commando, **ls**, om in het archive rond te kijken. Oudere versies (gedateerd voor December 1988) van de FTP daemon bevatten een security hole die het mogelijk maakte uit de beperkte FTP omgeving te ontsnappen. Inmiddels zijn er vele verbeterde versies verschenen, maar toch zijn er nog sites die een oude versie draaien, getuige de in sectie 7.1 te behandelen inbraakpoging.

3.5 NFS, NIS (YP), RPC, etc.

Deze Sun Microsystems producten die ter ondersteuning van data-sharing in een netwerk omgeving dienen, hebben in de laatste jaren enorm veel security holes opgeleverd; misschien kan men zelfs zeggen dat "SunOS vol zit/zat met security holes". Nu wàs SunOS natuurlijk nieuw en nog niet stabiel en ìs er de laatste tijd veel verbeterd, maar toch blijkt het nog steeds bijzonder moeilijk een acceptabel "veilig" systeem af te leveren. Ter illustratie: vorig jaar kreeg onze systeemgroep twee Sun-4 workstations draaiende onder het allernieuwste Sun operating system SunOS 4.1 te leen ter evaluatie. Na installatie van het systeem bleek tot onze grote verbazing dat de file `/etc/hosts.equiv` standaard geleverd wordt met enkel een '+' erin, hetgeen erop neerkomt dat alle systemen als *trusted host* beschouwd worden[8]. Als een systeembeheerder zelf zo dom is om er enkel een '+' in te zetten, dan verdient hij niet beter, maar in dit geval is de systeembeheerder (of de koper) slachtoffer van een onachtzaamheid van de leverancier!

Uiteraard was in dit eenvoudige voorbeeld een oplossing snel gevonden[9], maar het zou nog eenvoudiger geweest zijn als de leverancier het gewoon goed gedaan had; vooral voor die mensen die een systeem kopen om er mee te werken... ☹

In veel hedendaagse implementaties van NIS (vroeger YP geheten) zitten grote, niet te verhelpen security problemen die zich op twee manieren kunnen manifesteren: *onbedoeld onthullen van informatie* en *vertrouwen op onjuiste informatie.* Zonder in details te treden, komt het eerste probleem er op neer, dat iedereen die op de hoogte is van de NIS *domainname*, de NIS databases via het netwerk kan opvragen. Het tweede probleem werkt juist de andere kant op: het is mogelijk jezelf voor te doen als server voor de NIS databases, en als zodanig bijvoorbeeld een eigen versie van de password file aan te bieden als antwoord op een login-request.

[8] Zie voor een nadere toelichting hoofdstuk 5.

[9] Later bleek dat in het SunOS system administrators manual ook nog enige aandacht hieraan besteed werd.

4 De UNIX password organisatie

Hoewel het onderwerp "goed gebruik van passwords" nooit een *hot topic* zal worden en qua prestige nooit het niveau van "fouten in het operating system" zal halen, is het toch zo dat alles wat geïnvesteerd wordt in de beveiliging van een systeem voor niets geweest is, als het gebruikers-identificatie mechanisme met behulp van *passwords* niet deugt. In dit hoofdstuk zal ik uitleggen hoe het password coderingssysteem van UNIX in elkaar steekt.

User accounts worden op computersystemen beveiligd door middel van passwords. Uiteraard valt of staat het succes van dit mechanisme met de manier waarop gebruikers ermee omgaan: houden ze passwords geheim, is het password "moeilijk" genoeg, etc. Daarnaast is ook het operating system verantwoordelijk voor een goede omgang met passwords. Bijvoorbeeld is het van belang hoe (encryptie) en waar (verborgen of voor iedereen zichtbaar) passwords in het operating system opgeslagen worden.

Sommige systemen bieden tegenwoordig ook de mogelijkheid om de geldigheidsduur van een password vast te leggen en eisen dat na het vervallen van die geldigheidsduur een ander password gekozen wordt. Deze toevoeging heet *password-aging* en heeft mijns inziens vooral veel nadelen. Als men gebruikers dwingt vaak een nieuw password te kiezen, dan gaan zij makkelijker te onthouden passwords kiezen, of ze schrijven hun password op een briefje, dat ze vervolgens onder het toetsenbord van hun computer plakken. Gebruikers die altijd al een "moeilijk" te raden password gebruikt hebben, krijgen nu de lastige taak meerdere van dit soort passwords te bedenken. En wat belet een gebruiker continu te switchen tussen twee passwords?

4.1 Password codering in UNIX

In UNIX worden passwords gecodeerd opgeslagen, waarbij het gecodeerde password voor iedereen leesbaar blijft. Aangezien de gebruikte coderingsmethode een *one-way*[10] functie is, lijkt er niets aan de hand, maar schijn bedriegt en ik zal laten zien waarom. Uitgaande van een standaard password mechanisme zonder password-aging is een gecodeerd password[11] een string van 13 tekens lang, die er bijvoorbeeld als volgt uitziet:

```
mPAbQpI9.i9Sg
```

De password-codering moet garanderen dat het resultaat van de codering (encryptie) van een niet-gecodeerd password **altijd** dezelfde string van 13 tekens is: dit kan alleen als de coderings-sleutel bekend is. Op het moment dat bij het inloggen een password geverifiëerd wordt, zijn er dus twee dingen nodig: het niet-gecodeerde password en de coderings-sleutel (ook wel *crypt-key* of *salt* genoemd). Het eerste is eenvoudig te leveren, want dat is het password dat de gebruiker intikt. De crypt-key moet ergens anders vandaan komen en klaarblijkelijk een *public-key* zijn.

[10] *One-way* kan in deze context gelezen worden als "niet-omkeerbaar". Een functie is *one-way* als uit het resultaat van de functie, gegeven een bepaalde invoer, op geen enkele wijze de oorspronkelijke invoer verkregen kan worden.

[11] In feite wordt niet het password in gecodeerde vorm opgeslagen, maar wordt een blok data (64-bits) herhaaldelijk (25 keer) versleuteld aan de hand van het password en wordt het uiteindelijk resultaat daarvan opgeslagen.

In UNIX is het zo georganiseerd dat deze *public-key* de eerste twee tekens van het gecodeerde password vormt; in het voorbeeld dus de string "mP". Het gecodeerde password wordt dus verkregen door middel van:

```
crypted_password ::= CODEER(password, crypt_key);
```

Als we er in het voorbeeld vanuit gaan dat het niet-gecodeerde password "Geheim" was, dan kunnen we als volgt controleren of de gebruiker wel het goed password ingetikt heeft:

```
password_ok ::= ARE_EQUAL("mPAbQpI9.i9Sg", CODEER("Geheim", "mP"));
```

4.2 Het raden van passwords

De *one-way* eigenschap van de codering maakt het onmogelijk uit een gecodeerd password het oorspronkelijke password terug te vinden, maar, zoals uit sectie 4.1 moge blijken, is het wel mogelijk een password te achterhalen: niet door het te decoderen, maar door voldoende veel mogelijke passwords te coderen en te kijken of er een encryptie is, die de gecodeerde string oplevert. Nu is dit uiteraard een tijdrovend proces, maar met de hedendaagse snelle systemen zijn toch wel aardige resultaten te halen[12], zeker als je in ogenschouw neemt dat slechts de eerste 8 tekens van een password significant zijn.

De conclusie hieruit mag dan ook luiden dat het voor iedereen leesbaar opslaan van gecodeerde passwords tegenwoordig niet meer voldoet. In de nabije toekomst vormen zelfs "goede" passwords geen probleem meer voor de cracker. Enkele systemen bieden tegenwoordig een remedie tegen het publiekelijk opslaan van gecodeerde passwords: **shadow password** files. De nog steeds voor iedereen leesbare password file bevat geen gecodeerde passwords meer, maar enkel *verwijzingen* daarnaar. De gecodeerde passwords zijn opgeslagen op een (shaduw) plaats, waartoe alleen de super-user toegang heeft.

4.3 Veel gemaakte fouten met passwords

Accounts zonder password en/of group accounts komen nog op veel plaatsen voor en worden gebruikt om meerdere gebruikers toegang te geven tot een bepaald software produkt, bijvoorbeeld een database systeem. Veelal wordt de gebruiker dan in een beperkte omgeving neergezet en krijgt hij de beschikking over een commando interpreter met een zeer beperkte commando structuur, maar de ervaring leert dat ontsnappen uit die beperkte omgeving meestal eenvoudig is. Een betere oplossing zou zijn de betreffende software niet onder een gebruikersnaam onder te brengen, maar met zodanige permissies te installeren, dat een bepaalde *group-owner* er toegang tot heeft. Gebruikers die de software dan moeten gebruiken, kunnen door de systeembeheerder in die group geplaatst worden.

Volgens [Bran89] komen er vandaag de dag nog op ieder systeem zogenaamde "Joe's" voor; een "Joe" is een account waarvan de user-name gelijk is aan het password. Mijn experimenten met diverse password-files, afkomstig van diverse sites binnen de Rijksuniversiteit Utrecht, wezen uit dat dit inderdaad waar is: ik trof "Joe's" aan op vrijwel iedere onderzochte site.

[12] De UNIX encryptie is uitgevonden in een tijd waarin, als je zoiets toen zou willen proberen, je zeer waarschijnlijk niet lang genoeg zou leven om een succesvol resultaat te mogen meemaken. De momenteel snelste, nooit gepubliceerde implementatie van *crypt()* draait op een Cray supercomputer en levert meer dan 50.000 crypts per seconde.

Enkele leveranciers van hard- en software zijn uitermate bedreven in het leveren van systemen met een serie *default* passwords voor standaard user-names als *root*, *system*, *operator*, *diag*, *daemon*, *uucp* en *field*[13]. Dit lijkt op het eerste gezicht beter dan helemaal geen password, maar feit blijft dat systeembeheerders eerder geneigd zijn een account zonder password ontoegankelijk te maken, door bijvoorbeeld een '*' in het gecodeerde-password veld te zetten, dan al aanwezige passwords te veranderen. Toch is dit wat zou moeten gebeuren: de allereerste keer dat een nieuw systeem geboot wordt, dienen **onmiddellijk**, d.w.z. nog in *single-user* mode, **alle** default passwords veranderd te worden en dient een nieuw password voor de super-user geïnstalleerd te worden. Het is zeer belangrijk dat dit gebeurt **voordat** ook maar iemand anders toegang heeft tot het systeem.

[13]Overbekend is de combinatie user FIELD met password SERVICE op de DEC VAX/VMS systemen.

5 Security voor systeembeheerders

Systeembeheerders die security problemen willen voorkomen, moeten twee zaken nauwlettend in de gaten houden: de kwaliteit van het operating system (de systeem-software) en het gebruik van user accounts en passwords. Voor wat het operating system betreft, is het belangrijk tijdig te weten welke problemen er zijn en waar/welke patches beschikbaar zijn. Slecht gebruik van passwords kan men voorkomen door "goede" passwords af te dwingen en door gebruikers goed voor te lichten. Advies over het gebruiken van goede passwords is te vinden in hoofdstuk 6.

Naast het voorkomen van security problemen is nog iets erg belangrijk: weten dat je een security probleem hebt. Het hoeft niet ernstig te zijn als crackers af en toe wat met je systeem proberen, zolang je maar weet dàt het gebeurt. Het monitoren van een systeem kost veel tijd, maar met relatief eenvoudige middelen is het toch mogelijk datgene te zien wat belangrijk is: automatisering van het monitoren kan ervoor zorgen dat belangrijke meldingen van het operating system niet verdrinken in de overvloed aan onbelangrijke meldingen. Het kan ook handig zijn enkele keren per week een *samenvatting* van alle meldingen te genereren. De middelen die voor het monitoren van een systeem beschikbaar staan, zijn *accounting, daemon logging* en *consistency checks.*

Zoals opgemerkt ontstaan de meeste security problemen door een *security flaw* in de operating system software. In [Curr90] wordt duidelijk gemaakt hoe de security van een systeem verbeterd kan worden; het artikel is vooral gericht op beheerders van Sun systemen. Beheerders van HP9000 systemen kunnen vooral veel baat hebben bij [Hewl89]. Een meer algemene beschouwing is terug te vinden in [Bran89] en zeer aangename en illustratieve leesstof is te vinden in [Stol88] en [Stol89]. Pittige, zeer nuttige en bijzonder aan te bevelen boeken zijn [Wood85] en [Garf91]. Een kwalitatief goed algemeen boek over UNIX systeembeheer is [Neme89].

5.1 Tips voor de systeembeheerder ter verbetering van de security

Omdat het vrijwel onmogelijk is alle punten te beschrijven die voor verbetering van de security in aanmerking komen, zal ik in de vorm van een checklist kort ingaan op de belangrijkste **onderwerpen** en `files` die aandacht behoeven.

- **`/etc/passwd`**
  Dit is de standaard UNIX password file. Let erop dat *root* eigenaar is van deze file en dat de protectie `-rw-r--r--` is. Zie er verder op toe dat alleen *root* een user-id met waarde `0` heeft. Als NIS niet gebruikt wordt, verwijder dan alle regels die beginnen met een '+'. Is NIS wel actief, let er dan op dat de regels '`+::0:0:::`' en '+' alleen voorkomen op NIS client systemen, **niet** op de NIS master. Zoals reeds opgemerkt in sectie 4.3 is het belangrijk om alle default systeem passwords te veranderen, danwel deze accounts onbruikbaar te maken door bijvoorbeeld een '*' voor het gecodeerde password te zetten.

- **`/etc/hosts.equiv`**
  Deze file bevat de lijst van *trusted hosts.* Gouden regel is dat niet-lokale hosts **nooit** trusted behoren te zijn. Als deze file via NIS benaderd wordt, is het belangrijk dat er **nooit** een regel met enkel een '+' in staat: in dat geval worden namelijk **alle** hosts op het locale Ethernet (en het Internet) als trusted gezien!

- **/usr/lib/aliases**, soms ook **/etc/aliases**
  Deze file bevat *aliases* voor de **sendmail** mail software. Aliases die expanderen naar een programma kunnen gevaarlijk zijn, met name geldt dit voor de *decode* alias. Verwijder in deze file de regel:

  ```
  decode: "|/usr/bin/uudecode"
  ```

  en her-initialiseer de **sendmail** interne tabellen middels het commando:

  ```
  # newaliases
  ```

- **file protecties**
  Soms blijken files *world-writable* te zijn; voor files zoals **/.login**, **/.cshrc**, **/.rhosts**, **/etc/passwd** of **/etc/exports** is dat uiteraard vragen om problemen. Let ook op de protectie van de directories **/**, **/etc**, **/dev**, **/usr**, **/usr/adm**, **/bin** en **/usr/bin**.

  In een aantal gevallen is zelfs het voor iedereen leesbaar zijn van files gevaarlijk, bijvoorbeeld van devices als **/dev/[r]dsk/***, **/dev/mem** en **/dev/kmem**. SunOS systeembeheerders kunnen gebaat zijn bij het *sticky* bit en het *set-gid* bit op de directory protectie-modes.

  World-writable files zijn te localiseren met het commando:

  ```
  # find / -type f -perm -2 -exec ls -al {} \;
  ```

  Files die aan niemand toebehoren kunnen waarschijnlijk wel opgeruimd worden. Zoek dergelijke files met het commando:

  ```
  # find / -nouser -o -nogroup -print
  ```

- **environment variabelen, $PATH**
  Zet nooit en te nimmer de current directory '.' voorin je **PATH**, want dan is de kans wel enorm groot dat je in een val trapt. Voor de super-user geldt eigenlijk dat er eigenlijk helemaal geen '.' in de **PATH** variabele mag voorkomen: de super-user dient altijd een programma uit de huidige directory expliciet aan te roepen met **./program_name**. Ook het opnemen van zoiets als '**./bin**' is sterk af te raden. Een ideaal pad voor de systeembeheerder zou er als volgt uit kunnen zien (dit is een voorbeeld, de werkelijke vorm is systeem afhankelijk):

  ```
  PATH='/usr/local/bin:/bin:/usr/bin:/etc:/usr/etc:/usr/ucb'
  ```

- **netwerk services in /etc/inetd.conf (rlogin, telnet, ftp, tftp, ...)**
  Voor alle netwerk services geldt: stel buiten gebruik wat je niet nodig hebt en zorg ervoor dat datgene waar je niet buiten kunt een goed geconfigureerde recente versie is en limiteer gebruik daarvan tot *trusted* systemen. HP-UX biedt een nette mogelijkheid om services te beperken tot bepaalde systemen en/of (delen van) netwerken; op andere systemen is deze beperking mogelijk minder eenvoudig te bewerkstelligen. Wees erop bedacht dat het met oudere **tftp** versies mogelijk is alle files, die voor iedereen leesbaar zijn, ook vanaf een willekeurig ander systeem op het netwerk te lezen. Met een verkeerd geconfigureerde **tftp** service is het zelfs mogelijk **alle** files van een systeem te lezen.

- **set-uid programma's**
  Programma's die *set-uid root* zijn, vormen een potentiëel lek in de security. Als je zelf zo'n set-uid programma schrijft, ga dan uiterst zorgvuldig na of er niet een subtiel lekje in zit. Denk eraan dat de gebruikers twee componenten aanleveren die een potentiëel gevaar vormen voor ieder programma en met name de set-uid root programma's: input-data en de huidige environment[14]. Let vooral op het gebruik van de `system()` routine: gebruik daar altijd volledige padnamen èn herdefiniëer de belangrijkste environment variabelen, zoals `$PATH` en `$IFS`. Duidelijke voorbeelden van programmatuur zijn te vinden in [Wood85].

  Set-uid shell scripts vormen een intrinsiek gevaar voor de security en zijn op generlei wijze veilig te maken. Gebruik daarom **nooit** set-uid shell scripts en als het echt niet anders kan, verpak dergelijke scripts dan in een set-uid C programma of overweeg het script te herschrijven in *Perl* [Wall90].

  Voorkom zoveel mogelijk het gebruik van set-uid programma's en ga na of het gewenste resultaat ook bereikt kan worden met een *set-gid* programma. Algemeen kan gesteld worden dat "je nooit iets met set-uid moet oplossen, als het ook met set-gid kan".

- **secure terminals**
  'Secure terminals' is een nieuw mechanisme in een aantal recente UNIX systemen, o.a. SunOS 4.x, dat er kortweg op neer komt dat *root* niet mag inloggen op terminals die *niet* secure zijn. Helaas is het systeem initiëel zo geconfigureerd dat alle terminals (dus ook remote/network pseudo-terminals) als **secure** worden gezien. In SunOS biedt het niet-secure markeren van de systeem console tevens de veiligheid dat het niet meer mogelijk is het systeem in *single-user* mode te booten zonder het superuser password te kennen. Om eventuele misverstanden uit de weg te ruimen: een netwerk-terminal-connectie is eigenlijk pas *secure* te noemen als het key-word **secure** voor de betreffende terminal **ontbreekt** in `/etc/ttytab`...☹

- **"smart" terminals**
  Hedendaagse intelligente terminals hebben een zeer handige feature: een reeks characters, ingepakt in de juiste *escape-sequence*, zal de (te) intelligente terminal interpreteren als: "stuur deze string als commando naar de host computer". De host computer weet dan niet beter of de gebruiker op die terminal heeft deze string als commando ingetikt.
  Het commando "`mesg n`" (bijvoorbeeld in de `$HOME/.login` of de `$HOME/.profile`) voorkomt dat normale gebruikers zomaar tekst naar andermans terminal kunnen sturen, maar beschermt de gebruiker niet als dergelijke tekst via e-mail gestuurd wordt: aangeraden wordt dan ook een Mail User Agent te gebruiken die "vreemde" tekens uit een bericht filtert, bijvoorbeeld Elm.

[14] Voor kwaadwilligen zijn er een paar zeer creatieve mogelijkheden met de `IFS` environment variable.

- **user accounting**
  Dit kan varieren van het administreren van wie er ingelogd is (dit gebeurt minimaal op ieder UNIX systeem, zie de file `/etc/utmp`), via het bijhouden van deze gegevens over langere tijd (dit is meestal een optie, zie de file `/etc/wtmp`), tot het volledig administreren van ieder commando of zelfs van iedere uitgevoerde system-call. Dit laatste wordt *auditing* genoemd en is o.a. te vinden in HP-UX Trusted Systems.

- **failed logins en su gebruik**
  4.3BSD systemen loggen bad-logins en gelukte/mislukte *su* aanroepen via de *syslog* faciliteit. HP-UX doet dat anders, nl. mislukte logins gaan in `/etc/btmp` en su logging in `/usr/adm/sulog`. Als er veel mislukte logins achter elkaar optreden, dan is er zeer waarschijnlijk iemand bezig passwords te raden en wordt het tijd om uit te zoeken waar de terminal staat waarop dat geprobeerd wordt. Als het een netwerk pseudo-terminal blijkt te zijn, dan kan met `netstat` uitgevonden worden waar die connectie vandaan komt. Mocht het op een modem inbel-lijn gebeuren, dan wordt het aanmerkelijk moeilijker, maar in [Stol88] en [Stol89] wordt aangetoond dat er toch wel een paar mogelijkheden zijn.

- **plaats en tijd van last-login**
  Zeer nuttig is het als na het inloggen op een systeem de tijd en plaats van de vorige login getoond worden. Als het operating system deze service niet biedt, implementeer dan zelf zoiets. Als je als systeembeheerder ziet dat je ingelogd geweest bent op een tijdstip dat gewoon niet kan, dan is er een duidelijk probleem. Neem in dit geval de noodzakelijke maatregelen. Als het om een belangrijk systeem gaat, dat beschikt over gevoelige data, dan is een onmiddelijke *shutdown* en verder werken in *single-user mode* de enige juiste weg. Controleer alle filesystemen op files die gemodificeerd zijn sinds de laatste legitieme login en restore zo nodig een volledige backup. Vertel ook de gebruikers wat er aan de hand is en informeer hen over de veiligheids-procedure die uitgevoerd wordt.

- **set-uid en set-gid programma's**
  Wees bedacht op nieuwe *set-uid* en/of *set-gid* programma's; ga na wat ze doen, of dit mechanisme daarvoor wel nodig is en wie voor die programma's verantwoordelijk is. *Setuid* shell scripts zijn over het algemeen levensgevaarlijk, maar gelukkig is er public domain software voorhanden om op een veilige manier dit soort scripts te runnen. Programma's die *set-uid* root zijn, zijn eenvoudig als volgt te localiseren:

```
# find / -type f -a -user root -a -perm -04000 -exec ls -l {} \;
```

- **network daemon logging en syslog**
  Indien mogelijk, start network daemons zodanig op dat connecties aan die daemons (via de *syslog* faciliteit) gelogd worden. Het is altijd nuttig te weten waar al die mislukte logins via *telnet* of *rlogin* vandaan komen. In HP-UX worden network daemons vanuit de Internet daemon (inetd) gestart, die goede logging faciliteiten biedt. Om deze logging te verkrijgen, dient een modificatie gemaakt te worden in de file `/etc/netlinkrc`; kijk waar *inetd* opgestart wordt en verander die regel in:

```
/etc/inetd -l && /bin/echo "inetd (with logging)  \c"
```

- **netwerk IP routering**
  Toegang tot een systeem via een netwerk wordt vastgelegd in een *routeringstabel.* Deze tabel is te bekijken met het commando:

  ```
  # netstat -r
  ```

  Voor netwerk routering geldt dat je alleen die routes die je daadwerkelijk nodig hebt, in de tabel moet zetten. Gebruik waar mogelijk een *host-route* (`route add host ...`) in plaats van een *network-route* (`route add net ...`), om een zo duidelijk mogelijk toegangsgebied af te bakenen. Als je routeringstabel erg beperkt blijkt uit te vallen, ga dan over op *manueel* routeren en stel de RIP service (`routed` of `in.routed`) buiten gebruik.

- **trojan-horses en virussen**
  Helaas zijn dit vandaag de dag zeer populaire modegrillen en de kans dat UNIX hierdoor getroffen wordt is zeker niet denkbeeldig [Thom84, Witt87], gezien het enorme aanbod aan public domain software. Kijk uit met software van twijfelachtige afkomst; software die uit een moderated USENET newsgroup[15] komt, is meestal wel betrouwbaar. Voorzichtigheid is in het bijzonder geboden als je via FTP software van een archive-server haalt waarbij je opvalt dat de directories *world-writable* zijn: iedereen kan immers een wijziging in de daar aanwezige files aangebracht hebben.

  Uiteraard is wel iets tegen virussen te doen, nl. het bewaken van de consistentie van de data op je filesystemen, met name op het `/` en `/usr` filesysteem. Als je denkt door een virus getroffen te zijn, kun je veel werk voorkomen indien je kunt garanderen dat deze beide filesystemen niet geïnfecteerd zijn. HP-UX Trusted Systems biedt hiervoor de zgn. *Production Description Files*, maar uiteraard is het niet al te moeilijk om zelf iets dergelijks in elkaar te sleutelen (denk aan CRC checks).

Uit deze checklist zal duidelijk geworden zijn, dat veel security problemen voortkomen uit verkeerd geconfigureerde systemen. Alleen het feit al dat het blijkbaar mogelijk is, een checklist tegen deze problemen op te stellen, doet vermoeden dat één en ander toch te automatiseren zou moeten zijn. Welnu, er is inderdaad goede software, die kan helpen bij het opsporen van potentiële security problemen, voorhanden: **COPS**. COPS is een software pakket ontwikkeld door Dan Farmer `<df@cert.sei.cmu.edu>` en wordt gedistribueerd via het CERT (zie appendix A voor de verkrijgbaarheid van deze software), dat iedere systeembeheerder minstens eenmaal per maand zou moeten draaien op zijn systemen.

[15] Voorbeelden zijn `comp.sources.unix`, `comp.sources.misc` en `comp.sources.x`

6 Security voor gebruikers

Op het eerste gezicht lijkt het kiezen van "goede" passwords de enige wezenlijke bijdrage die gebruikers kunnen leveren, om tot een verbetering van de veiligheid van een computersysteem te komen, maar niets is minder waar. Vaak is het kraken van een account van een gewone gebruiker, en dus het verkrijgen van toegang tot een systeem, de eerste stap tot een verder kraken van een systeem. Andere gebruikers of de systeembeheerder hebben veelal wel weer een weggetje geplaveid voor de misbruiker om tijdelijk wat meer bevoegdheden te krijgen. Daarom in dit hoofdstuk een behandeling van de overige punten waar de gebruikers op dienen te letten. De aandachtig lezer zal in dit hoofdstuk enige overlapping met het vorige hoofdstuk ontdekken.

6.1 Tips voor gebruikers ter verbetering van de security

Naast de systeembeheerder kan ook de gebruiker maatregelen nemen die de beveiliging van een systeem verbeteren. De volgende checklist kan hierbij behulpzaam zijn:

- **`$HOME/.login`, `$HOME/.profile`, `$HOME/.cshrc`, `$HOME/.logout`**
  Deze zgn. *dot–files* van normale gebruikers dienen nooit voor iedereen writable te zijn. Zelfs het readable zijn is af te raden, want dan is wel erg eenvoudig uit te vinden welke programma's de gebruiker telkenmale bij het inloggen uitvoert. Die programma's zijn dan uitgesproken kandidaten om er een *virus* in te planten.

- **`$HOME/.rhosts`**
  Deze file legt (samen met de `/etc/hosts.equiv` file) vast welke combinaties van host-name/user-name als **trusted** gezien worden. Toegang tot een systeem verkrijgen zonder eerst een password in te moeten tikken, is bijzonder handig, maar men zou eens voor zichzelf na moeten gaan of men echt alle in de `.rhosts` file genoemde systemen, en dus alle gebruikers op die systemen, het volledige vertrouwen wil schenken. Algemeen geldt dat men niet-lokale hosts maar beter niet kan vertrouwen.

- **`$HOME/.netrc`**
  De `.netrc` file wordt door de library functie *rexec(3N)* en het commando *ftp* gebruikt en dient ervoor om de gebruiker te ontlasten van het intikken van de loginnaam/password combinatie. Een entry uit de `.netrc` file ziet er als volgt uit:

  ```
  machine XXX login YYY password ZZZ
  ```

  Passwords staan in **niet-gecodeerde** vorm in deze file, hetgeen ingaat tegen de in sectie 6.2 te noemen regels betreffende het gebruik van passwords. Algemeen geldt: maak **nooit** gebruik van een `.netrc` file.

- **let op plaats en tijd van last-login**
  Als deze service door het systeem geboden wordt, let er dan op of het niet een "onmogelijk" inlog-tijdstip aangeeft. Raak in twijfelgevallen ("Zondagnacht?? Dat kan niet, toen was ik...") niet in paniek, maar meld je bange vermoedens onmiddelijk bij de systeembeheerder.

- **environment variabelen, $PATH**
  Zet nooit en te nimmer de current directory '.' voorin je PATH, want dan loop je de kans met open ogen in een val te trappen welke, als hij goed is opgezet, niet te achterhalen zal zijn. Zo je al door mocht hebben dat er iets niet helemaal in de haak is, dan nog zal het vreselijk moeilijk zijn uit te vinden wat de val beoogde te bewerkstelligen. Mocht je toch in zo'n val lopen, waarschuw dan zo spoedig mogelijk de systeembeheerder.

- **umask**
  De waarde van *umask* bepaalt met welke protectie-mode nieuwe files gecreëerd worden. Hij is default meestal 022, hetgeen betekent dat nieuwe files met protectie-mode `-rw-r--r--` gecreëerd worden, en dus voor iedereen leesbaar zijn. In de praktijk zal liever de waarde 027 (resulterend in de protectie-mode `-rw-r-----`) of soms zelfs 077 (resulterend in de protectie-mode `-rw-------`) gehanteerd worden.

6.2 Een goede omgang met passwords: diverse aanbevelingen

Zoals reeds in sectie 4.2 is gesuggereerd, worden tegenwoordig veel passwords gekraakt door er maar een heel "woordenboek" van mogelijkheden tegenaan te gooien. Ook hebben passwords vaak een logisch verband met de eigenaar, en iemand die de achtergrond van de betreffende persoon kent is meestal wel in staat met enige slimheid de juiste te raden. In [Curr90] wordt aangetoond dat variaties op de login-naam, de voor- en achternaam van de gebruiker en een lijst van zo'n 1800 algemene voornamen voldoende zijn om op vrijwel ieder systeem binnen twee à drie dagen **50%** van de passwords te achterhalen!

Een deel van de strategie van de ontwerper van de *Internet worm* [Eich89, Seel89, Spaf88] was erop gericht passwords te kraken, teneinde op andere systemen in te kunnen breken. In eerste instantie probeerde de *worm* eenvoudige keuzes zoals login-naam, voor- en achternaam van de gebruiker, permutaties daarvan, etc. Vervolgens werd er een interne woordenlijst van 432 woorden op losgelaten. Tenslotte werd het on-line woordenboek `/usr/dict/words` geprobeerd. Alle password aanbevelingen die hieronder gedaan worden, vormen een betrouwbare remedie tegen deze drie strategieën.

Als gebruiker kan men zich tegen het raden van passwords wapenen door een password te kiezen dat niet in een woordenboek voorkomt en geen enkele verband met jezelf heeft. Als men zijn password zo moeilijk maakt dat het redelijkerwijs met slim *raden* niet meer te achterhalen is, dan blijft er voor de cracker niets anders over dan met brute-force alle mogelijke combinaties van letters, cijfers en punctuaties te proberen. Zelfs op snelle systemen met een bijzonder snelle uitvoering van de codeer algoritme, zal dit toch meer dan 10000 jaar rekentijd vergen.
De volgende richtlijnen kunnen behulpzaam zijn bij het kiezen van een goed password:

- gebruik **niet** je login-naam of een permutatie daarvan, noch aparte uitvoeringen als dubbel, omgekeerd, met hoofdletters, etc.,
- gebruik **niet** je voor- of achternaam, noch een afgeleide daarvan,
- gebruik **niet** de naam van je lievelingshuisdier, man/vrouw of kind,
- gebruik **niet** dingen die eenduidig met jezelf verbonden zijn, zoals telefoonnummer, kenteken, fiscaal nummer, merknaam van je auto/fiets, straatnaam waar je woont, favoriete vrijetijdsbesteding, favoriete stripheld, etc.,

- gebruik **niet** een password dat enkel uit cijfers of allemaal dezelfde letters bestaat of dat een simpele afgeleide van de keyboard-layout vormt (denk aan *qwerty*),
- gebruik **niet** woorden die in een woordenboek, in spellinglijsten, of in andere woordenlijsten voorkomen,
- gebruik **niet** een password van minder dan 6 tekens,
- schrijf passwords **niet** ergens op,
- gebruik **wel** kleine letters en hoofdletters door elkaar,
- gebruik **wel** een password dat niet-alfabetische tekens bevat, zoals cijfers, leestekens, spaties e.d.,
- gebruik **wel** een password dat je eenvoudig kunt onthouden,
- gebruik **wel** een password dat je snel kunt typen, zodat iemand die over je schouder meekijkt het niet kan volgen.

Als men al deze richtlijnen in ogenschouw neemt, lijkt het wel enorm moeilijk geworden nog een goed password te verzinnen. Daarom nog de volgende tips die daarbij kunnen helpen:

- kies twee korte bestaande woorden en koppel die middels een speciaal teken aan elkaar, zodat er èn een niet-bestaand woord ontstaat èn er een niet-alfabetisch teken in voorkomt. Voorbeelden zijn:

  "boter" en "vis" ⇨ "boter+vis"
  "boek" en "put" ⇨ "put,boek"
  "hond" en "bal" ⇨ "hond/bal"

- neem een woord dat je eenvoudig kunt onthouden (en dat best in een woordenboek mag voorkomen) en vervang daarin een bepaalde vaste letter door een speciaal teken. Voorbeelden:

  "fietsbel" ⇨ "fiets!el"
  "aureool" ⇨ "aure//l"

- kies je favoriete regel uit een song van je favoriete (pop)groep en neem van ieder woord de eerste letter, bijvoorbeeld "Six lanes of traffic, three lanes moving Slow..."[16] leidt tot het password "SlottlmS".

Gebruikers die over meerdere accounts beschikken, mogelijk doordat ze op diverse plaatsen werkzaam zijn, staan voor het dilemma of ze overal hetzelfde danwel overal een ander password moeten nemen. Ik denk dat in het algemeen onderstaande richtlijnen aangehouden moeten worden.

[16] Uit "Telegraph Road" door Mark Knopfler en Dire Straits, van het album *Love over Gold.*

Passwords mogen identiek zijn dan en alleen dan als:

1. de systemen logisch equivalent zijn, bijvoorbeeld deel uitmaken van een *pool* van workstations, **en**
2. de systemen elkaar vertrouwen (ga dit na bij je systeembeheerder), **en**
3. de systemen door dezelfde systeembeheerder(s) beheerd worden.

Passwords dienen verschillend te zijn als:

1. de systemen door verschillende systeembeheerders beheerd worden, **of**
2. de systemen een verschillend niveau van security hebben, **of**
3. de systemen een verschillend operating system draaien.

Onthoud ook dat een cracker die beschikt over een goede kennis van cryptografie, voordeel heeft van het beschikken over twee verschillend gecodeerde versies van één en hetzelfde password.

7 Recente gevallen en de gevolgen

Nu de veel voorkomende problemen behandeld zijn, wordt het tijd om duidelijk te maken dat we niet volkomen paranoïde zijn, maar dat genoemde maatregelen echt nodig zijn. Ik zal een aantal gevallen behandelen uit de periode tussen april 1990 en april 1991. Uiteraard zal ik de slachtoffers niet met naam en toenaam noemen, behalve als onze systeemgroep zelf slachtoffer was. Voor een goed begrip is het nodig te weten dat ik over een **poging** praat als de beoogde kraak niet succesvol was. Een **kraak** is een succesvolle poging waarbij aanzienlijke schade berokkend is.

7.1 Poging 1: Vakgroep Informatica, Rijksuniversiteit Utrecht

Toen ik op 20 april 1990 eens relatief vroeg (rond 07.45 uur) op m'n werk verscheen, bleek onmiddelijk dat we onder attack lagen van iemand die probeerde via diverse netwerk services (telnet, ftp, tftp, smtp, rlogin, finger, nntp) binnen te komen op al onze HP9000 UNIX systemen. Vreemd genoeg vonden deze pogingen plaats vanaf *twee* systemen, die zo te zien niets met elkaar te maken hadden.

Systeem A was een *universitair* systeem uit Canada; systeem B een *government* systeem ergens uit de VS. Enig snel speurwerk, dat ik vanachter mijn workstation kon uitvoeren, wees uit dat op systeem B iemand via een FTP netwerk-connectie vanaf systeem A als *SYSTEM* ingelogd was. Bingo! Dat kon geen toeval zijn...☺
De login-naam *SYSTEM* wees overduidelijk op een VAX/VMS machine, waarvan het beheerdersaccount misbruikt werd: mogelijk gold voor dat account nog het default password dat er door de overijverige leverancier ingezet was.

Enige conversatie via e-mail met de manager van EUnet/NLnet[17] was voor deze manager aanleiding onmiddelijk het CERT[18] in te schakelen. Later op de dag ontving ik bericht dat het CERT er inderdaad in was geslaagd de beheerders van systeem B ervan te overtuigen dat er iets volkomen fout zat bij hen. Daarmee werd in ieder geval dit lek gedicht.

In dit geval heeft een goede configuratie van de verschillende netwerk daemons de poging in de kiem gesmoord en werd voldaan aan de regel dat men netwerk services waar men niet buiten kan zodanig moet configureren dat enkel vertrouwde systemen er gebruik van kunnen maken. Tevens heeft dit bewezen hoe belangrijk een goede logging van netwerk activiteiten is: immers dankzij deze logging is ergens anders op de wereld een lek in de security gedicht.

[17] EUnet is de Europese tak van het DARPA Internet; NLnet is de Nederlandse tak van EUnet.

[18] Het Computer Emergence Response Team ([Peth90]) is een team bestaande uit ruim 100 netwerk en systeem-software experts verspreid over de gehele wereld. CERT werd direct na het gevecht tegen de Internet worm opgericht door DARPA. Het CERT Coördination Center is gehuisvest in het Software Engineering Institute, Carnegie Mellon University. Het CERT beoogt o.a. het (helpen) opsporen en dichten van security holes en het voorkomen van "rampen" zoals de Internet worm. Het CERT is via e-mail bereikbaar als **cert@cert.sei.cmu.edu** en telefonisch, 24-uur per dag, op nummer **+1-412-268-7090**.

7.2 Kraak 1: een onderzoekscentrum in Nederland

Op de systemen van een onderzoekscentrum in Nederland werden in begin juli 1990 sporen van crackers gevonden. Nadat duidelijk was geworden dat men via een modem-connectie binnen was gekomen, en gebruik gemaakt had van het account van iemand van de *system staff*, was er reden genoeg voor een grondige controle van het gehele systeem. Na enige tijd werd duidelijk dat de crackers nieuwe ingangen hadden weten te creëren, simpelweg door de `.rhosts` files van een aantal gebruikers aan te passen. Gelukkig had het onderzoekscentrum ruim voldoende UNIX kennis in huis om de problemen snel de baas te kunnen worden.

Aangenomen dat de persoon wiens account misbruikt was, zijn password netjes geheim gehouden heeft, blijkt hier weer hoe belangrijk het is om een moeilijk password te gebruiken. Zijn het niet de eigen spullen die men wil beschermen, dan nog is men het ethisch verantwoord tegenover andere gebruikers op het systeem en het netwerk.

7.3 Kraak 2: een universiteit in Nederland

Deze gecompliceerde kraak, die in mei 1990 plaatsvond, was er een die wel wat weg had van wat in [Stol89] beschreven wordt. De connecties liepen over enorm veel netwerken, zowel in Nederland als in de Verenigde Staten. Ik ben niet volledig op de hoogte van alle details, maar het volgende zal zeker duidelijk maken hoe ernstig zo'n kraak kan zijn.

De kraak werd gepleegd gebruik makend van vele netwerken. De connectie begon zeer waarschijnlijk in een Nederlandse huiskamer met als gereedschap een simpele microcomputer, een modem en een telefoontoestel. Via een telefoonlijn werd ingebeld op een computersysteem (mogelijk een soort terminal-server waarvoor geen account benodigd was), vanwaar men naar de VS ging. Vanuit een systeem in de VS (waar blijkbaar al een account gekraakt was) liep de verbinding weer terug naar Nederland en uiteindelijk via DECnet naar het rekencentrum van de universiteit die slachtoffer in deze zaak was. Duidelijk moge zijn dat het opsporen van de crackers in een dergelijke situatie moeilijk, zo niet bijna onmogelijk is.

Eenmaal binnen op een systeem van het rekencentrum hebben de crackers zich niet beperkt tot rondkijken, maar hebben ze zich als vandalen gedragen. Op de systemen van het rekencentrum werd een dusdanige schade aangericht dat men een aantal systemen vanaf *scratch* opnieuw heeft moeten installeren. Gelukkig voerde het rekencentrum een goed backup beleid zodat de schade, afgezien van enkele dagen werk, nog binnen de perken bleef. Een faculteit was minder gelukkig: daar werd een volledige cluster Apollo systemen zo grondig schoongeveegd dat er niets meer overbleef om nog mee te werken. Helaas werd er op die faculteit geen goed beleid ten aanzien van backups gevoerd, want de meest recente backup bleek een **jaar** oud te zijn. Resultaat was dus dat die faculteit een jaar werk (scripties, proefschriften, onderzoeksresultaten, etc.) in rook heeft zien opgaan.

Later is hetzelfde rekencentrum nogmaals slachtoffer geworden van een succesvolle kraak: deze keer werden alle Ultrix machines volkomen platgelegd. In beide gevallen verkreeg men ingang tot het systeem via een DECnet verbinding. In DECnet zijn een aantal commando's geïmplementeerd die het mogelijk maken files van een remote host te lezen, ook als je daar geen account hebt. De remote host kan deze commando's weigeren, maar helaas blijkt dat de leverancier (DEC) de verkeerde default configuratie gekozen heeft: standaard accepteert het systeem deze commando's wel...☹

7.4 Kraak 3: diverse overheidsinstellingen in de VS

In februari 1991 kwam de Rijksuniversiteit Utrecht (RUU) op een negatieve manier in de belangstelling via de nationale pers en haalde de RUU zelfs de *Achter het Nieuws* uitzending op de TV. De filmploeg van de VARA liet twee crackers zien die demonstreerden hoe ze vanuit huis, gebruik makend van een systeem van de RUU, toegang hadden weten te krijgen tot onder meer systemen van de NASA in de VS.

De crackers maakten gebruik van de RUU inbel-service via een terminal-server (de zgn. Develnet-switch), vanwaar toegang gekozen kon worden tot een beperkt aantal systemen binnen de RUU. Het uitgekozen systeem, een VAX van de faculteit Letteren, bleek de *DECnet gateway-functie* te bieden. Deze functie maakt het mogelijk **zonder** in te loggen aan het systeem kenbaar te maken dat je niet met het systeem zelf wilt werken, maar via DECnet een ander systeem wilt gaan gebruiken. Daarmee was voor de crackers de weg naar de rest van de wereld geplaveid...[19]

In een bericht in de New York Times van zondag 21 april 1991 kwam John Markoff nog eens terug op deze kwestie. Dit bleek aanleiding tot een hevige discussie in de USENET newsgroup `comp.risks`.

[19] Op moment van de uitzending was dit lek al bekend en door de systeembeheerders gedicht.

8 Conclusies

Uit mijn experimenten zijn een paar dingen duidelijk geworden die ik hier kort samenvat. Allereerst is gebleken dat de *Internet worm* ook vandaag nog (bijna 3 jaar na dato), zeer succesvol kan zijn. Hoewel mijn ervaring zich niet uitstrekt buiten het RUU netwerk, maar ik ook geen enkele reden heb om aan te nemen dat de situatie elders anders zou zijn, ben ik ervan overtuigd dat een versie van de *worm* die aangepast is, aan wat er recentelijk aan security holes boven water is gekomen, een behoorlijke dreun kan uitdelen. Als deze worm een wat kwaadaardiger karakter heeft, dan zal de toegebrachte schade aanzienlijk zijn.

Tevens is gebleken dat er nog steeds **zeer onzorgvuldig** met passwords omgesprongen wordt. Ik ben behoorlijk geschrokken van het aantal passwords dat ik binnen zeer korte tijd (variërend van enkele minuten tot een uur) heb weten te kraken. Bij het kraken heb ik gebruik gemaakt van exact dezelfde woordenlijst die de *Internet worm* gebruikte, en daarmee bleek het toch mogelijk een klein percentage passwords snel te achterhalen. Met een uitgebreidere woordenlijst en een intelligenter programma bleek het op enkele plaatsen mogelijk binnen een tijdsbestek van enkele uren **25%** van de passwords te achterhalen, waaronder het *root* password.

Het echte probleem voor een systeembeheerder schuilt in de complexiteit van "doing it right". Je kunt niet zomaar een systeem uitpakken, inpluggen en ermee aan de slag. Je moet wel een zeer ervaren systeembeheerder zijn, wil je weten wat je moet corrigeren, en dat geldt niet alleen voor items die op de security betrekking hebben. Er was bijv. een leverancier die een operating system upgrade leverde die resulteerde in een protectie-mode 777 (`drwxrwxrwx`) voor alle systeemdirectories. Bij hoeveel instanties heeft de systeembeheerder tijd om het USENET news netwerk te monitoren op zoek naar de laatste patches? Hoeveel sites hebben volledige toegang tot alle net-faciliteiten? Hoe zou je moeten weten dat er een probleem is, als je geen toegang tot USENET news hebt? Hmm, misschien zouden hard- en software leveranciers kunnen vertellen dat er problemen zijn; iets wat nogal eens (bewust) nagelaten wordt...☹

9 Nawoord

Ik hoop in dit artikel duidelijk gemaakt te hebben wat de gevolgen kunnen zijn, als er slecht omgesprongen wordt met de middelen die voorzien in de security van computersystemen. Ook hoop ik dat dit artikel enigszins een bijdrage in de verbetering van de security van UNIX systemen moge vormen.

Verder wil ik een woord van dank uitspreken aan de UNIX systeembeheerders van de diverse faculteiten/vakgroepen van de Rijksuniversiteit Utrecht, die mij meestal met, maar soms ook zonder medeweten (☺), behulpzaam zijn geweest bij het uitvoeren van mijn security experimenten. Ook ben ik gelukkig met de veelal positieve reacties die ik kreeg, wanneer ik telefonisch bekend maakte welke problemen ik had geconstateerd.

Ik dank met name prof. dr. J. van Leeuwen voor zijn interesse in dit onderwerp en zijn vele waardevolle suggesties.

Tenslotte dank ik Henk Penning, Lex Wolters en Fred Appelman voor de geleverde bijdragen en de vele uren besteed aan het doorlezen van voorlopige versies van dit artikel.

Referenties

[Bran89] Russell L. Brand, *Coping with the Threat of Computer Security Incidents — A Primer from Prevention through Recovery*, IEEE Compcon seminar handout, CERT-0.6, 1990.

[Curr90] David A. Curry, *Improving the Security of Your UNIX System*, SRI International, Information and Telecommunications Sciences and Technology Division, Technical Report ITSTD-721-FR-90-21, Menlo Park, CA, 1990.

[Denn89] Peter J. Denning, *The Internet Worm*, American Scientist, March-April 1989, pp. 126-128.

[Donn89] Donn Seeley, *Password Cracking: A Game of Wits*, Communications of the ACM, volume 32, issue 6, June 1989, pp. 700-703.

[Eich89] Mark W. Eichin and Jon A. Rochlis, *With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988*, Technical Report, Massachussets Institute of Technology, 1989.

[Eise89] Ted Eisenberg, David Gries, Juris Hartmanis, Don Holcomb, M. Stuart Lynn and Thomas Santoro, *The Cornell Commission: On Morris and the Worm*, Communications of the ACM, volume 32, issue 6, June 1989, pp. 706-709.

[Garf91] Simson Garfinkel and Gene Spafford, *Practical UNIX Security*, O'Reilly and Associates, Inc., Sebastopol, CA 95472, 1991.

[Hewl89] Hewlett-Packard Company, *HP-UX System Security; HP9000 series 300/800 Computers*, HP Part Number 92453-90029, 1989.

[Neme89] Evi Nemeth, Garth Snyder and Scott Seebass, *UNIX System Administration Handbook*, Prentice-Hall, Englewood Cliff, NJ 07632, 1989.

[Peth90] Richard D. Pethia and Kenneth R. van Wyk, *Computer Emergency Response - An International Problem*, CERT Coördination Center, Software Engineering Institute, Carnegie Mellon University, 1990.

[Seel89] Donn Seeley, *A Tour of the Worm*, Technical Report, Dept. of Computer Science, University of Utah.

[Spaf88] Eugene H. Spafford, *The Internet Worm Program: An Analysis*, Purdue Technical Report CSD-TR-823, Dept. of Computer Science, Purdue University, 1988.

[Spaf89] Eugene H. Spafford, *Crisis and Aftermath*, Communications of the ACM, volume 32, issue 6, June 1989, pp. 678-687.

[Stol88] Clifford Stoll, *Stalking the Wily Hacker*, Communications of the ACM, volume 31, issue 5, May 1988, pp. 484-497.

[Stol89] Clifford Stoll, *The Cuckoo's Egg - Tracking a Spy Through the Maze of Computer Espionage*, Doubleday, New York, 1989.

[Thom84] Ken Thompson, *Reflections on Trusting Trust*, Communications of the ACM, volume 27, issue 8, August 1984, pp. 761-763.

[Wall90] Larry Wall and Randal L. Schwartz, *Programming Perl*, O'Reilly and Associates, Inc., Sebastopol, CA 95472, 1990.

[Witt87] Ian H. Witten, *Computer (In)security: Infiltrating Open Systems*, ABACUS, volume 4, issue 4, Summer 1987.

[Wood85] Patrick H. Wood and Stephan G. Kochan, *UNIX System Security*, Hayden Books, Indianapolis, 1985.

A Verkrijgbaarheid van genoemde papers

Een aantal van de op pagina 25 genoemde referenties zijn beschikbaar via anonymous FTP van onze archive-server **archive.cs.ruu.nl**. Login als **anonymous**, met als password je e-mail adres, getuige de volgende voorbeeld sessie. Vergeet niet binary-mode te gebruiken!

```
$ ftp archive.cs.ruu.nl
Connected to sol.cs.ruu.nl.
220 sol FTP server (Utrecht University, Dept. of Computer Science) ready.
Name (sol.cs.ruu.nl:edwin): anonymous
331 Guest login ok, enter your e-mail address as password.
Password (sol.cs.ruu.nl:anonymous): edwin@cs.ruu.nl
230 -Guest login ok, access restrictions apply.
230 Local time is: Tue Jul 30 14:30:23 1991
ftp> verbose
Verbose mode off.
ftp> binary
ftp> cd SECURITY
ftp> get worm-mit.ps.Z
ftp> get cops-1.02.tar.Z
ftp> bye
221 Goodbye.
$
```

De genoemde referenties zijn in de volgende files terug te vinden:

Referentie	*File naam*	*Formaat*
[Bran89]	sec-primer.tar.Z	LaTeX
[Curr90]	security-doc.tar.Z	troff/postscript
[Eich89]	worm-mit.ps.Z	postscript
[Peth90]	security.response.ps.Z	postscript
[Seel89]	worm-utah.ps.Z	postscript
[Spaf88]	worm-purdue.ps.Z	postscript

B Trademarks

De onderstaande trademarks en produktnamen (in alfabetische volgorde) worden in dit artikel gebruikt:

Apollo is a subsidiary of Hewlett-Packard.

DEC, DECnet, VAX, VMS and Ultrix are trademarks of Digital Equipment Corporation.

HP-UX is Hewlett-Packard's implementation of UNIX.

NFS is a trademark of Sun Microsystems, Inc.

NIS (also known as Yellow Pages or YP) is Sun's Network Information Service.

PostScript is a registered trademark of Adobe Systems, Inc.

SunOS is Sun Microsystems' implementation of UNIX.

Ultrix is a trademark of Digital Equipment Corporation.

UNIX is a registered trademark of AT&T in the USA and other countries.

VMS is a trademark of Digital Equipment Corporation.

— ☼ — ☼ — ☼ —