TCP WRAPPER, een tool voor het bewaken van netwerkactiviteit

Wietse Venema

Wiskunde en Informatica
Technische Universiteit Eindhoven
wietse@wzv.win.tue.nl

ABSTRACT

Dit verhaal presenteert aan de hand van praktijkvoorbeelden een aantal netwerkmonitoring technieken die ik heb ontwikkeld voor het vroegtijdig detecteren van verdachte netwerkactiviteiten. Dit alles zonder enige aanpassing in reeds bestaande systeemsoftware en configuratiefiles, en op een zodanige manier dat normale gebruikers geen hinder ondervinden. Het gereedschap mag zich verheugen in een behoorlijke populariteit, en wordt zelfs aanbevolen door organisaties zoals CERT.

1. De jacht op het huisdier.

Het verhaal begint een kleine twee jaar geleden met een kraker die regelmatig op TUE-systemen inbrak en op magische wijze telkens superuser wist te worden. Dat zou op zichzelf nog niet een ramp zijn, maar hij had ook een heel hinderlijke voorliefde voor het commando:

```
rm -rf /
```

Voor niet UNIX-ingewijden: dit heeft ongeveer hetzelfde effect als het `format`-commando onder MS-DOS. Na zo'n calamiteit kon met behulp van van backups de directe schade meestal hersteld worden, maar de niet-materiele schade was natuurlijk veel ernstiger. En de backups konden niet voorkomen dat zo nu en dan iemand toch een grote hoeveelheid werk verloor.

Bij gebrek aan 100% waterdichte bewijzen kan ik natuurlijk geen namen noemen, maar wij hadden wel uiterst sterke aanwijzingen. In de wandelgangen noemden we de kraker "ons huisdier". Voor insiders moet dit voldoende informatie zijn.

2. Door de kraker bespioneerd.

Nadat een inbraak was geconstateerd was het onmogelijk om sporen te onderzoeken. Die waren immers door de `rm -rf` acties volledig uitgewist. Ik ontdekte echter dat de kraker ons voortdurend over het netwerk bespioneerde. Hij deed dat onder andere door verbinding te maken met o.a. onze `finger` netwerkservice. Daarmee kon hij zien wie er op de systemen waren ingelogd. Dat kon hij vrijwel ongemerkt doen omdat dit soort netwerkservices geen wachtwoorden vereist, en omdat gewoonlijk niet wordt vastgelegd wie van dit soort services gebruik maakt.

De voor de hand liggende reactie zou zijn geweest om de `finger` en soortgelijke netwerkservices af te sluiten. Het leek me in dit geval productiever om de netwerkservices juist te handhaven en te proberen om er achter te komen vanuit welke systemen de kraker ons zat te bespioneren.

3. Een typische UNIX TCP/IP netwerkimplementatie in vogelvlucht.

Om het probleem en de gevonden oplossing inzichtelijk te maken moet ik eerst in het kort samenvatten hoe een typische UNIX-implementatie van de TCP/IP netwerkfaciliteiten is georganiseerd. Mensen die hiermee op de hoogte zijn zullen het mij hopelijk vergeven dat ik zo hier en daar wat vereenvoudigingen heb aangebracht.

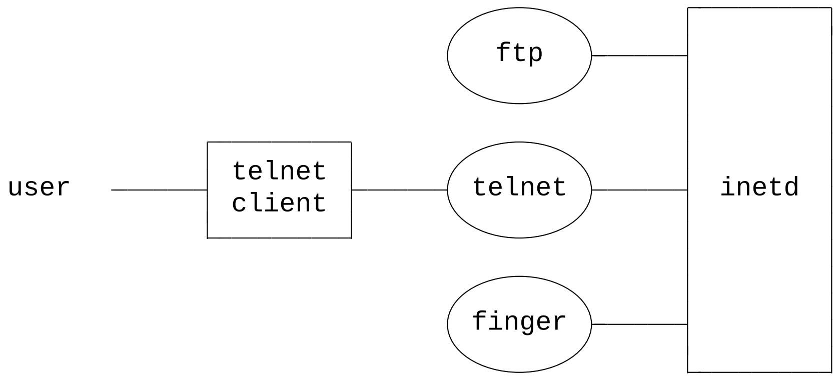
Nagenoeg alle TCP/IP-toepassingen zijn gebaseerd op het *client-server* model. Bijvoorbeeld, als een gebruiker met het `telnet`-commando contact maakt met een host, dan wordt op dat laatste systeem een *telnet server* gestart die de gebruiker met een `login` proces verbindt. Enkele voorbeelden zijn gegeven in tabel 1.

client	server	toepassing
`telnet`	`telnetd`	remote login
`ftp`	`ftpd`	file transfer
`finger`	`fingerd`	wie is ingelogd
`systat`	`systatd`	wie is ingelogd

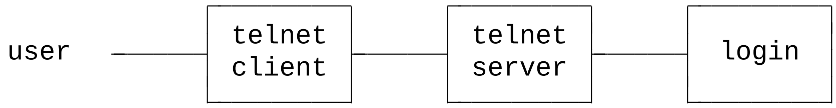
Tabel 1. Voorbeeld van een aantal TCP/IP client-server combinaties.

Net zoals de `ftp` en `telnet` gebruikerscommandos meestal door verschillende programma´s worden geimplementeerd, zijn dus ook de bijbehorende netwerkservers in het algemeen geimplementeerd als aparte programmas.

De op dit moment gangbare aanpak is om slechts een daemon proces op binnenkomende netwerkverbindingen te laten wachten. Wanneer een verbinding wordt gemaakt start deze daemon (gewoonlijk `inetd` genaamd) het bijbehorende serverprogramma op en gaat vervolgens wachten op nieuwe binnenkomende verbindingen.



Figuur 1. Het `inetd` proces luistert naar de `ftp, telnet` etc. netwerk poorten en wacht op binnenkomende connecties. In de figuur is aangegeven dat er zojuist door een gebruiker met een `telnet` clientprogramma een verbinding met de telnet poort is gemaakt.



Figuur 2. Het `inetd` proces heeft een `telnet` serverproces gestart dat de verbinding verder afhandelt; `inetd` wacht nu op nieuwe binnenkomende connecties. Het `telnet` serverproces verbindt de gebruiker met een `login` proces.

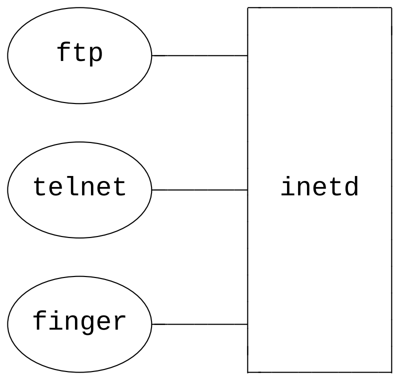
4. De "tcp wrapper" wisseltruuk.

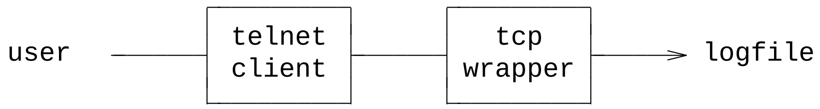
Om de naam te achterhalen van de host van waaruit de kraker ons bespioneerde zou dus aanpassing van de netwerksoftware vereist zijn. Op het eerste gezicht waren de vooruitzichten niet zo gunstig:

- o De TUE had geen licentie voor de source code van de UNIX-versies die op onze systemen werden gebruikt. De source code zelf hadden we overigens ook niet.
- o De Berkeley netwerk source code, waarvan uiteindelijk alle UNIX netwerk software is afgeleid, was weliswaar beschikbaar, maar het zou een onbekende hoeveelheid werk kosten om die geschikt te maken voor onze UNIX systemen. Leveranciers plegen onder het mom van "verbeteringen" allelei obscure wijzigen aan te brengen.

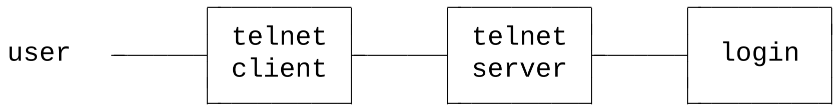
Gelukkig kon ik volstaan met een oplossing die geen enkele wijziging in systeemsoftware vereiste, en die bovendien op al onze UNIX systemen bleek te werken. Daartoe maakte ik gebruik van een eenvoudige wisseltruuk: installeer op de plaats van elk network serverprogramma een triviaal programmaatje, en verplaats de originele network servers naar een andere directory. Wanneer nu een verbinding binnenkomt start het *inetd* proces het triviale programmaatje op in plaats van de echte netwerkserver. Het triviale programmaatje logt de naam van de remote host en start vervolgens het echte serverprogramma op.

Zo konden we de netwerkactiviteit in de gaten houden terwijl er geen enkele systeemsoftware of configuratiefile hoefde te worden aangepast.





Figuur 3. Op de plaats van de oorspronkelijke `telnet` server is het *tcp wrapper* programma geinstalleerd. Dit programma logt de naam van de remote host naar een logfile.



Figuur 3. Figuur 4. Het *tcp wrapper* programma heeft de echte `telnet` server gestart en doet nu verder niet meer mee. Vanuit de gebruiker gezien is dus niets veranderd.

De eerste versie van de *tcp wrapper* bestond uit slechts een paar regeltjes code die ik zorgvuldig had overgenomen uit een bestaande netwerk daemon source. En omdat het geen enkele informatie uitwisselde met het remote clientproces, kon dezelfde *tcp wrapper* versie worden gebruikt voor allerlei verschillende netwerkservices.

Ondanks het feit dat ik de *tcp wrapper* op slechts een dozijn systemen kon installeren was het meteen een succes. Figuur 5 geeft een voorbeeld.

De kraker maakte veelvuldig gebruik van de `systat` en `finger` services. Daarmee kon hij zien zien wie er op een systeem waren ingelogd. Verder maakte de kraker zo nu en dan een `telnet` verbinding. Vermoedelijk om na een enkele inlogpoging meteen de verbinding te verbreken, zodat er geen "repeated login failure" boodschap op de systeemconsole zou verschijnen.

Terwijl de kraker dacht ons ongemerkt te bespioneren konden we vanaf nu dus precies zien waar hij bezig was. Dit was een hele vooruitgang vergeleken met vroeger, toen we pas wat merkten als het te laat was, namelijk als hij weer eens een `rm -rf` had gedaan.

Aanvankelijk was mijn vrees dat we bedolven zouden worden onder de logs, en dat we tussen alle ruis de gewenste informatie niet zouden kunnen vinden. Gelukkig bleek het vrij eenvoudig om de kraker te herkennen:

```
May 21 14:06:53 tuegate: systatd: connect from monk.rutgers.edu
May 21 16:08:45 tuegate: systatd: connect from monk.rutgers.edu
May 21 16:13:58 trf.urc: systatd: connect from monk.rutgers.edu
May 21 18:38:17 tuegate: systatd: connect from ap1.eeb.ele.tue.nl
May 21 23:41:12 tuegate: systatd: connect from mcl2.utcs.utoronto.ca
May 21 23:48:14 tuegate: systatd: connect from monk.rutgers.edu

May 22 01:08:28 tuegate: systatd: connect from HAWAII-EMH1.PACOM.MIL
May 22 01:14:46 tuewsd:  fingerd: connect from HAWAII-EMH1.PACOM.MIL
May 22 01:15:32 tuewso:  fingerd: connect from HAWAII-EMH1.PACOM.MIL
May 22 01:55:46 tuegate: systatd: connect from monk.rutgers.edu
May 22 01:58:33 tuegate: systatd: connect from monk.rutgers.edu
May 22 02:00:14 tuewsd:  fingerd: connect from monk.rutgers.edu
May 22 02:14:51 tuegate: systatd: connect from RICHARKF-TCACCIS.ARMY.MIL
May 22 02:19:45 tuewsd:  fingerd: connect from RICHARKF-TCACCIS.ARMY.MIL
May 22 02:20:24 tuewso:  fingerd: connect from RICHARKF-TCACCIS.ARMY.MIL

May 22 14:43:29 tuegate: systatd: connect from monk.rutgers.edu
May 22 15:08:30 tuegate: systatd: connect from monk.rutgers.edu
May 22 15:09:19 tuewse:  fingerd: connect from monk.rutgers.edu
May 22 15:14:27 tuegate: telnetd: connect from cumbic.bmb.columbia.edu
May 22 15:23:06 tuegate: systatd: connect from cumbic.bmb.columbia.edu
May 22 15:23:56 tuewse:  fingerd: connect from cumbic.bmb.columbia.edu
```

Figuur 5. Enkele van de de eerste kraker connecties die door de *tcp wrapper* werden waargenomen. Elke connectie is gelogd met: tijdstip, naam van de locale host, de gevraagde service, en de naam van de remote host. Hieruit blijkt dat de kraker niet alleen gebruik maakte van terminal servers (zoals `monk.rutgers.edu`), maar ook dat hij toegang had tot allerlei militaire (`.MIL`) en universitaire (`.EDU`) systemen.

- Hij was vaak in de nacht actief, wanneer er weinig andere activiteit is.
- Hij maakte altijd in betrekkelijk korte tijd een aantal connecties naar verschillende systemen op de TUE. Door zijn activiteit te spreiden over verschillende systemen dacht hij misschien minder op te vallen.
- Omdat ik de logfiles van verschillende systemen samenvoegde was het juist extra gemakkelijk om te zien wanneer de kraker in de lucht was.
- Niemand anders gebruikte de `systat` service.

De activiteiten van de kraker staken dus ruimschoots boven de "ruis" uit.

In het bovenstaande voorbeeld is een van de `systat` connecties afkomstig van een systeem binnen de TUE: `ap1.eeb.ele.tue.nl`, dat deel uitmaakt van een ring van Apollo workstations. Aangezien de kraker de enige was die van de `systat` service gebruik maakte was het duidelijk dat hij nog steeds op de TUE actief was. Ik heb dan ook onmiddelijk de beheerder gewaarschuwd. Helaas was deze moeite voor niets: een week later werden alle disks van die Apollo ring gewist. Aangezien de laatste backup ruim anderhalf jaar oud was was de schade aanzienlijk.

5. Open terminalservers considered harmful.

Zoals uit het voorbeeld blijkt, opereerde de kraker graag vanaf open terminalservers. Dit soort systemen is niet voor niets populair bij krakers. Als je een normale usercode gebruikt om op een ander systeem in te breken ben je vrij eenvoudig te achterhalen. De meeste multi-user systemen houden immers logfiles bij met login tijden en andere informatie. Door de logs op het gekraakte systeem te vergelijken met de logs van het systeem van waaruit de inbraak werd gepleegd kun je er meestal vrij snel achter komen om welke usercodes het gaat.

In het geval van open terminalservers ben je veel moeilijker te achterhalen.

- o Terminalservers vereisen zelden een wachtwoord.
- o Terminalservers zijn vaak direct bereikbaar vanuit het openbare telefoon netwerk.
- o Terminalservers kunnen vaak als springplank worden gebruikt om van het ene systeem naar het andere over te springen, zodat de oorspronkelijke hostnaam niet in de loginaccounting te zien is.
- o Terminalservers houden geen logfiles bij, zodat je niet kunt nagaan waarvoor ze zijn gebruikt.

Kortom, terminalservers met wereldwijde internet access zijn ideaal omdat je in volstrekte anonimiteit kunt opereren. In feite zijn terminalservers nog erger dan reguliere computersystemen met slechte wachtwoorden, omdat de laatse tenminste nog enige loginaccounting bijhouden en omdat accounts kunnen worden afgesloten.

Ik zou een lange tirade kunnen houden over de bezwaren van publiek toegankelijke terminalservers met wereldwijd internet access. In het kort komt het hierop neer: in geval van inbraken vanuit een persoonlijke usercode op een reguliere host kan de betreffende user code worden opgespoord en afgesloten; in het geval van inbraken vanuit een terminalserver is dat niet mogelijk. Mijn advies is dan ook om terminalservers te blokkeren voor interlocaal verkeer, zodat ze door krakers niet als uitvalsbasis kunnen worden gebruikt.

6. Eerste uitbreiding: access control.

Het is duidelijk dat we weinig zouden hebben aan sporen die doodliepen op open terminalservers. De logische stap was dan ook om connecties vanuit dat soort systemen te gaan weigeren, zodat de kraker alleen nog kon opereren vanuit usercodes op reguliere computersystemen. De hoop was dat de kraker meer sporen zou achterlaten, zodat we wat meer van hem te weten zouden komen.

Aldus werd een eerste versie van access control in de *tcp wrapper* ingebouwd. Telkens wanneer een terminalserver in de logfiles opdook werd die onmiddelijk aan onze kant geblokkeerd, en werd de beheerders van die terminalservers gevraagd hetzelfde aan hun kant te doen. Met dat laatste hadden we overigens lang niet altijd succes. Figuur 6 geeft een idee hoe onze access control files er op een bepaald moment uitzagen.

```
/etc/hosts.allow:

    in.ftpd: ALL

/etc/hosts.deny:

    ALL: terminus.lcs.mit.edu hilltop.rutgers.edu monk.rutgers.edu
    ALL: comserv.princeton.edu lewis-sri-gw.army.mil
    ALL: ruut.cc.ruu.nl 131.211.112.44
    ALL: tip-gsbi.stanford.edu
    ALL: tip-quada.stanford.edu
    ALL: s101-x25.stanford.edu
    ALL: tip-cdr.stanford.edu
    ALL: tip-cromemaa.stanford.edu
    ALL: tip-cromembb.stanford.edu
    ALL: tip-forsythe.stanford.edu
```

Figuur 6. Voorbeeld van access control files. De eerste file beschrijft welke service/host combinaties altijd zijn toegestaan. In dit voorbeeld worden `ftp` connecties altijd geaccepteerd, ongeacht de herkomst van de verbinding.

De tweede file beschrijft welke resterende service/host combinaties worden geweigerd. In dit voorbeeld worden services geweigerd aan een reeks open terminalservers, ongeacht het type van de gevraagde service.

7. Een koekje van eigen deeg: de kraker bespioneerd.

Nu we de kraker de mogelijkheid hadden ontnomen om ons aan te vallen vanuit terminalservers, bleef voor hem niets anders over dan om direct vanuit gekraakte usercodes te opereren. Het leek dan ook nuttig om te proberen uit te vinden welke usercodes hij daarvoor gebruikte.

Ik draaide snel iets in elkaar dat bij connecties vanuit "verdachte" systemen een `finger` en `systat` naar die systemen terugdeed. Na alle gespioneer door de kraker zouden we hem dus eindelijk eens een koekje van eigen deeg kunnen geven.

In de loop van de tijd identificeerde ik met mijn *reverse fingers* diverse usercodes waarop de kraker had ingebroken. Uiteraard stelde ik steeds de beheerders op de hoogte van het probleem. Een copietje ging dan steeds naar CERT[1], zodat zij op de hoogte bleven van de vorderingen. Het was natuurlijk zaaks om vooral nooit mail te sturen naar het het gekraakte systeem zelf, aangezien de kans groot was dat de kraker het dan onder ogen zou krijgen.

```
Jan 30 04:55:09 tuegate: telnetd: connect from guzzle.Stanford.EDU
Jan 30 05:10:02 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:17:57 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:18:24 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:18:34 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:18:38 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:18:44 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:21:03 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:24:46 tuegate: systatd: connect from guzzle.Stanford.EDU
Jan 30 05:27:20 svin01:  fingerd: connect from gloworm.Stanford.EDU
Jan 30 05:33:33 svin01:  telnetd: connect from guzzle.Stanford.EDU
Jan 30 05:33:38 svin01:  telnetd: connect from guzzle.Stanford.EDU
Jan 30 05:33:41 svin01:  telnetd: connect from guzzle.Stanford.EDU
Jan 30 05:33:50 svin01:  ftpd:    connect from guzzle.Stanford.EDU
Jan 30 05:33:58 svin01:  fingerd: connect from math.uchicago.edu
Jan 30 05:34:08 svin01:  fingerd: connect from math.uchicago.edu
Jan 30 05:34:54 svin01:  fingerd: connect from math.uchicago.edu
Jan 30 05:35:16 svin01:  fingerd: connect from guzzle.Stanford.EDU
Jan 30 05:35:36 svin01:  fingerd: connect from guzzle.Stanford.EDU
```

Figuur 7. Een burst netwerkactiviteit, vrijwel allemaal vanuit Stanford.

```
Wed Jan 30 05:10:08 MET 1991

[guzzle.stanford.edu]
Login name: adrian                      In real life: Adrian Cooper
Directory: /u0/adrian                   Shell: /phys/bin/tcsh
On since Jan 29 19:30:18 on ttyp0 from tip-forsythe.Sta
No Plan.
```

Figuur 8. Een *reverse finger* resultaat behorende bij een van de bovenstaande connecties. Er was slechts een gebruiker ingelogd, dus we hadden een schot in de roos.

Uit de voorbeelden in figuren 7 en 8 blijkt dat de kraker op het systeem `guzzle.Stanford.EDU` was ingelogd als `adrian`. Hij kwam vanaf `tip-forsythe.Stanford.EDU`, een terminalserver die aan een modem bank is gekoppeld. Op het eerste gezicht was dit resultaat dus weinig hoopgevend, maar de afloop was bepaald anders.

---

1. Computer Emergency Response Team, een organisatie die in het leven is geroepen na het incident met de *internet worm*, november 1988.

In dit geval verwees CERT mij door naar Stephen Hansen. Hij logde al enige tijd alle sessies van adrian. De kraker voelde zich kennelijk thuis in Stanford, want hij heeft van daaruit gedurende enige maanden lang diverse andere systemen aangevallen. Ik heb alle logs van Stanford bekeken, en ze geven een uitstekend inzicht in hoe de kraker te werk ging.

- o De kraker was uitstekend op de hoogte van bugs in systeemsoftware. Gelukkig kende hij ze niet allemaal.
- o Hij controleerde uiterst zorgvuldig alle file- en directoryprotecties op mogelijke zwakheden in de beveiliging.
- o Als hij een systeem wist binnen te komen was dat vrijwel altijd omdat gebruikers zwakke wachtwoorden gebruikten. Door maar genoeg systemen te proberen kwam hij altijd wel ergens binnen.

Met zijn kennis en geduld wist de kraker in de meeste gevallen superuser privileges te verkrijgen.

Op Stanford wist de kraker diverse malen bijna superuser te worden. Op zulke momenten was er dan altijd, heel toevallig, een storing in een disk of iets dergelijks, zodat hij het karwei nooit heeft kunnen afmaken.

Een van de systemen die de kraker vanuit Stanford op de korrel nam was `research.att.com`, een gateway van de AT&T Bell laboratories. Die activiteiten bleven bepaald niet onopgemerkt. De mensen van AT&T vonden het zelfs wel interessant en installeerden een complete namaak omgeving waarin ze de kraker zijn gang lieten gaan. Ondertussen werd hij uitvoerig bespioneerd. Een en ander is beschreven door Bill Cheswick in een artikel genaamd "An evening with Berferd", hetgeen onlangs werd gepresenteerd op een bijeenkomst van de Amerikaanse USENIX vereniging van UNIX gebruikers [1].

De kraker is overigens nooit gearresteerd. Hij was gewoon een jaar te vroeg met zijn activiteiten. Het is jammer dat in plaats van dit destructieve individu twee veel minder schadelijke "data reizigers" de twijfelachtige eer hadden om als eerste Nederlandse hackers te worden gearresteerd.

8. Booby traps.

Tot nu toe had ik onze network daemons geinstrumenteerd om netwerk connecties te rapporteren, waarna ik zelf de interessante informatie moest selecteren. Daarbij maakte ik weliswaar gebruik van uiterst complexe pattern matching procedures, maar de uiteindelijke beoordeling of een systeem al dan niet moest worden opgenomen in de lijst van verdachte systemen bleef toch een handmatig proces.

Wat ik eigenlijk wilde was een soort "boobytrap", een apparaat dat automatisch alarm zou slaan in geval van verdachte activiteit, en dat bij normaal gebruik niets van zich zou laten horen. Hiervoor kon ik de gewone `telnet` en dergelijke services niet gebruiken.

Een geschikte netwerkservice bleek de trivial file transfer (`tftp`) service. Dit protocol vereist geen wachtwoord, en wordt voornamelijk gebruikt tijdens het booten van een diskloos werkstation, een X terminal, of van dedicated netwerk hardware, om de systeem software over het netwerk in te laden.

Tot voor een paar jaar was *tftp* een security probleem omdat je daarmee iedere file op het systeem kon lezen, dus ook de UNIX password file. De wachtwoorden worden weliswaar met een gemuteerd DES algoritme versleuteld, maar het algoritme is niet geheim, en met een goede woordenlijst en een slimme password kraker kun je gemakkelijk zwakke wachtwoorden vinden. Gelukkig voorzien de meeste huidige *tftp* implementaties in een restrictie op welke files toegankelijk zijn. Er zijn echter nog heel wat systemen die verouderde software draaien.

In de loop van de tijd had ik al diverse *tftp* connecties gezien vanaf systemen die niets met de TUE hadden te maken. Ook in dit geval leek het me nuttiger om de service niet af te sluiten, maar om de reverse finger faciliteit op niet-locale tftp connecties toe te passen.

Daartoe was een kleine aanpassing in de de *tcp wrapper* nodig. Tot nog toe werkte mijn *reverse finger* code immers uitsluitend op basis van "verdachte" hostnamen. Nu moest er ook naar de naam van de service worden gekeken, want ik wilde niet voor elke interlocale netwerkverbinding een *reverse finger* doen.

Ongeveer elke twee maanden loopt er wel iemand in de fuik. In zo’n geval stuur ik een mailtje naar de contactpersoon van de betreffende organisatie (nooit naar het gekraakte systeem zelf) plus een copietje naar CERT.

```
/etc/hosts.allow:

    in.tftpd: LOCAL

/etc/hosts.deny:

    in.tftpd: ALL: /usr/ucb/finger -l @%h 2>&1 | /usr/ucb/mail wswietse
```

Figuur 9. Entries in de aangepaste access control files om de tftp service van een boobytrap te voorzien. De eerste file legt vast dat `tftp` connecties vanaf locale systemen altijd zijn toegestaan.

De tweede access control file bepaalt dat alle overige `tftp` connecties altijd worden geweigerd. Het resultaat van een *reverse finger* wordt mij per email toegestuurd. In het bovenstaande commando wordt `%h` vervangen door de naam van de remote host.

Het volgende is een voorbeeld van een verdachte tftp connectie:

```
Jan  4 18:58:28 svin02 tftpd: refused connect from E40-008-8.MIT.EDU
Jan  4 18:59:45 svin02 tftpd: refused connect from E40-008-8.MIT.EDU
Jan  4 19:01:02 svin02 tftpd: refused connect from E40-008-8.MIT.EDU
Jan  4 19:02:19 svin02 tftpd: refused connect from E40-008-8.MIT.EDU
Jan  4 19:03:36 svin02 tftpd: refused connect from E40-008-8.MIT.EDU
Jan  4 19:04:53 svin02 tftpd: refused connect from E40-008-8.MIT.EDU
```

Aangezien tftp is gebaseerd op een datagram protocol zien we elke 17 seconden een herhaling van het geweigerde request. De precieze lengte van het retry interval is implementatie afhankelijk.

Ook in dit geval lieten de *reverse finger* resultaten niets te raden over:

```
Login       Name               TTY Idle    When            Office
mvscott  Mark V Scott           p0       Sat 12:46  14S-134       x3-6724
aparkin  Adam P Arkin           p2   15 Sat 12:38  56-207        253-6517
cuomo    Kevin M Cuomo          p3   24 Sat 12:10  LIN-KB-157   181-0772
mwsmith  Merrill W Smith        p6   13 Sat 11:33  7-238          x8-5596
vastettn Victoria A Stettner    r0  18d Mon 14:51  E53-231       x3-2722
```

Er was maar een gebruiker die in aanmerking kwam: `mvscott`. Alle anderen hadden gedurende een kwartier of langer niets gedaan. Blijkens de gedetailleerde *finger* resultaten was ingebroken vanuit `cnam.cnam.fr`, een systeem in Frankrijk.

```
Login name: mvscott                     In real life: Mark V Scott
Office: 14S-134,  x3-6724
Directory: /mit/mvscott                 Shell: /bin/csh
On since Jan  4 12:46:44 on ttyp0 from cnam.cnam.fr
12 seconds Idle Time
No Plan.
```

Uit de reactie van de mensen in Frankrijk bleek het te gaan om iemand die bij hun was ingebroken vanuit een terminalserver bij NASA:

```
hyper1    ttyp3    sdcds8.gsfc.nasa Sat Jan  4 17:51 - 20:47  (02:55)
```

(de volledige hostnaam is `sdcds8.gsfc.nasa.gov`). Al met al werd er driemaal over de Atlantische oceaan gegaan: vanuit een onbekende plaats, via een terminalserver bij NASA, via een systeem in Frankrijk, via een systeem bij het MIT, naar onze systemen.

9. Conclusies.

De *tcp wrapper* is een nuttig gereedschap gebleken om de toegang via netwerken te bewaken. Verscheidene versies van het programma zijn via het internet verspreid. Het programma mag zich in een behoorlijke populariteit verheugen. Het wordt vrijwel dagelijks van onze ftp server opgehaald naar alle delen van de wereld. Daarnaast wordt het beschikbaar gesteld door organisaties zoals CERT, die het in haar lijst van aanbevolen software heeft opgenomen.

- o Het vereist geen enkele wijziging van reeds bestaande systeemsoftware of configuratiefiles. Men hoeft dus geen guru te zijn om het te kunnen installeren.
- o Met de meeste UNIX varianten kan het zonder enige wijziging worden geinstalleerd. UNIX systemen vormen veruit de meerderheid van de op het internet aangesloten systemen.
- o Het belemmert legale gebruikers niet bij het gebruik van computer systemen.
- o Het doet niet mee bij het afhandelen van de eigenlijke netwerkservice, dus de kans op bugs is uiterst klein.
- o Het kan worden gebruikt voor zowel TCP (connection oriented) als UDP (datagram oriented) services, voor zover die door een centraal proces zoals de `inetd` worden opgestart.
- o Het biedt bescherming tegen zogenaamde "name server spoofing", waarbij een systeem de naam van een andere host probeert aan te nemen. Deze beveiliging is belangrijk voor netwerkservices zoals `rsh` en `rlogin` omdat die gebruik maken van authenticatie op basis van hostnaam. Wanneer een hostnaam niet blijkt te kloppen wordt de verbinding domweg verbroken.
- o De optionele access control faciliteit kan nuttig zijn om al te open systemen af te schermen. Iets dergelijks kan ook met netwerk routers worden gerealiseerd, maar dat kan vertragend werken op het overige netwerk verkeer. Bovendien maken routers zelden melding van ongewenste verbindingen.
- o De mogelijkheid tot het opzetten van "boobytraps" op bepaalde netwerk services zoals tftp, of op connecties vanuit verdachte systemen, bleek nuttig om tijdig te worden gewaarschuwd voor verdachte activiteit.

De *tcp wrapper* is slechts een van de voorzieningen die ik op onze systemen heb aangebracht. Al met al beschikken we over een redelijk effectief instrumentarium om krakers te detecteren en om ze in de gaten te houden. Het is dan ook niet verwonderlijk dat de TUE in krakerskringen de naam heeft redelijk beveiligd te zijn.

10. Referenties.

[1] Cheswick, W.R. *An Evening with Berferd, in Which a Cracker is Lured, Endured, and Studied*. Proceedings of the Winter USENIX Conference, (San Francisco), January 1992. Tevens beschikbaar voor anonymous ftp op:
`research.att.com: /dist/berferd.ps`.

[2] De tcp wrapper source is verspreid via de USENET news groep comp.sources.misc en is beschikbaar voor anonymous ftp op:
`cert.sei.cmu.edu: /pub/network_tools/tcp_wrapper.shar`,
`ftp.win.tue.nl: /pub/security/log_tcp.shar.Z`.