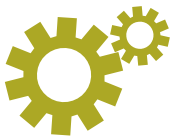
hakin9

Analyse des Netzwerkverkehrs

Bartosz Przybylski

Das *hakin9* Magazin, Software-Wydawnictwo, ul. Piaskowa 3, 01-067 Warschau, Polen *de@hakin9.org*



Techniken

Analyse des Netzwerkverkehrs

Bartosz Przybylski

Schwierigkeitsgrad

Wenn Sie ein Netzwerk verwalten, können Sie sicher sein, dass es früher oder später zum Ziel eines Angriffs wird. Sie sind allerdings im Stande, seine Erfolgschancen vielleicht nicht völlig vernichten, zumindest aber stark herabsetzen. Die Methoden dazu sind viele: Deaktivieren von Diensten, Firewalls, bis hin zu den IDS. Möglicherweise erweist sich dennoch die Fähigkeit, guten Verkehr im Netzwerk vom bösen zu unterscheiden, als das Wichtigste.

Pcap ist eine der üblichsten Bibliotheken, mit denen Netzwerkverkehr aufgezeichnet werden kann. Sie bietet einen feindetailierten Zugriff auf die einzelnen ISO/OSI-Schichten und steht für mehrere Betriebssysteme (mehr dazu in Konrad Malewskis Artikel in *hakin9* 6/2005 (13)) und in verschiedenen Programmiersprachversionen zur Verfügung.

Hammer, Schraubenzieher, Sniffer, oder die Analysewerkzeuge

Wenn wir uns mit der Netzwerkanalyse befassen sollen, sind wir auf bestimmte Tools angewiesen, die uns die Arbeit wesentlich leichter machen. Sehen wir uns zuerst Sniffer an.

Ethereal

Ethereal ist eins der bekanntesten Tools der Netzwerkanalyse. Es bietet eine ganze Reihe von Funktionalitäten, die man dabei nützlich finden kann. Die zwei wichtigsten Merkmale des Projekts sind: die Aufzeichnung des Netzwerkverkehrs in verschiedenen Formaten und die grafische Oberfläche. Obwohl das letztere nicht direkt für die Analyse erforderlich ist, macht es die Arbeit angenehmer und bequemer.

Tcpdump

Tcpdump ist ein weiterer sehr guter Sniffer. Seine Autoren sind es, die auch die Pcap-Bibliothek entwickelt haben. Für das Programm stehen mehrere inoffizielle Frontends zur Wahl,

In diesem Artikel erfahren Sie...

- Wie der Verkehr in einem Netzwerk analysiert werden kann;
- Wie Angriffsversuche in der Analyse aufgespürt werden können;
- Wie solche Versuche gesperrt werden können.

Was Sie vorher wissen/können sollten...

- Kenntnisse der Grundlagen von Rechnernetzen (das ISO/OSI-Modell);
- Kenntnisse der Linux-Shell.

Listing 1. *Authentizitätsprüfung von zwei Dateien (aut.sh)*

```
#!/bin/sh
if [ -z $2 ]; then
echo "Usage: $0 authentic_file file_for_check";
exit
if

md5sum $1 | cut -c1-32 > /tmp/f1.cksum
md5sum $2 | cut -c1-32 > /tmp/f2.cksum
sha1sum $1 | cut -c1-40 >> /tmp/f1.cksum
sha1sum $2 | cut -c1-40 >> /tmp/f2.cksum

res=`/usr/bin/cmp /tmp/f1.cksum /tmp/f2.cksum`

if [ -z "$res" ]; then
echo "File is authentic"
else
echo "File is not authentic"
fi

rm /tmp/f1.cksum /tmp/f2.cksum
```

entworfen wurde es jedoch mit shellbasierter Arbeit im Sinn.

Unser Arsenal

Hier ist die Liste der Anwendungen und Skripte, deren wir uns bei der Analyse des Netzwerkverkehrs bedienen:

- *capinfos* (ein Teil des ethereal-Pakets);
- *tcpdstat*;
- *zonk.pl* (ein einfaches Skript für Netzwerkadministratoren, vom Autor des Artikels entwickelt;
- ein paar eigene Skripte.

Garantierte Authentizität

Wenn die Ergebnisse der durchgeführten Analyse als Anklagebeweise gegen den Angreifer dienen sollen, müssen wir gewährleisten, dass die Authentizität der Datei mit der Aufzeichnung des Netzwerkverkehrs und der untersuchten Datei belegt werden kann.

Wichtig ist, dass das Erstellungsdatum der Originaldatei, die als Hauptbeweis fungiert, dem Angriffsdatum möglichst nahe ist. Um das zu gewährleisten, kopieren wir die Aufzeichnung in ein separates Verzeichnis und stellen für die Datei und das Verzeichnis die Zugriffsberechtigung *schreibgeschützt* (*read-only*) mit den folgenden Befehlen ein:

```
mkdir ~/analyze
cp ./traffic.cap ~/analyze
chmod 444 ~/analyze/traffic.cap
~/analyze/
```

Nachdem wir die Kopie gesichert haben, die als Fundament unserer Beweise dient, müssen wir noch ihre Authentizität belegen, und zwar mit dem einfachen Skript aus Listing 1.

Das Skript kann sich auch beim Beweisen der Authentizität des aufgezeichneten Verkehrs als nützlich erweisen. Es lohnt sich außerdem, Prüfsummen für die Originaldatei zu führen, was noch überzeugender wirkt.

Listing 2. *Ausgabe des capinfo-Programms*

```
$ capinfo traffic.cap
1   File name: traffic.cap
2   File type: libpcap (tcpdump, Ethereal, etc.)
3   Number of packets: 1194
4   File size: 93506 bytes
5   Data size: 213308 bytes
6   Capture duration: 342.141581 seconds
7   Start time: Thu Jun 23 14:55:18 2005
8   End time: Thu Jun 23 15:01:01 2005
9   Data rate: 623.45 bytes/s
10  Data rate: 4987.60 bits/s
11  Average packet size: 178.65 bytes
```

Netzwerkanalyse

Nun kommen wir zu unserer zentralen Aufgabe, das heißt der Analyse des abgefangenen Verkehrs. Dazu brauchen wir einige Grundinformationen über den Verkehr, die wir mit dem Programm *capinfo* aus dem *ethereal*-Paket ansammeln.

Sehen wir uns Listing 2 an und überlegen wir, was für Informationen uns *capinfo* liefern kann.

Die erste Zeile (*File name*) mit dem Dateinamen lassen wir aus. In der zweiten ist das Dateiformat angegeben. Die Beispielsdatei ist im *pcap*-Format gespeichert. Ein anderes übliches Speichersystem für Netzwerkverkehr ist *Microsoft Network Monitor x.x*, wo x.x die Bibliothekversion ist. Die häufigste Fassung ist 2.x (natürlich sind es nicht die einzigen Formate für Verkehrsaufzeichnungen).

Die nächste Ausgabezeile (3) gibt die Zahl der Pakete an, die während der Sniffing-Sitzung im Netzwerk übertragen wurden. Darauf folgen die Dateigröße (4) und der Umfang des aufgezeichneten Verkehrs (5). Danach stehen: die genaue Sitzungsdauer (6), das Start- (7) und Enddatum (8) der Sitzung, der durchschnittliche Durchsatz in Bytes/s (9) und in Bits/s (10), und die letzte Zeile (11) gibt die Durchschnittsgröße eines Pakets an.

Bereits an dieser Etappe der Analyse können wir auf eventuellen illegalen Verkehr folgern (vor allem in großen Unternehmensnetzen). Wenn innerhalb eines kurzen Zeitraums (bei einem relativ kleinen *Capture duration* Wert) untypisch viele Pakete bzw. eine hohe Durchschnittsgröße der Pakete auftreten,

können wir die Nutzung von P2P-Software verdächtigen. Dies muss allerdings nicht wahr sein – solch ein Verkehrsmuster kommt auch beispielsweise bei Softwareaktualisierungen vor.

Der nächste Schritt ist, den Verkehr genauer zu untersuchen, diesmal auf der Paketebene. Was uns interessiert, sind der Protokolltyp und die Paketgröße. Darüber hinaus erfahren wir mehr über die Leitungsauslastung. Dabei bedienen wir uns des modifizierten *tcpdstat*-Programms von Dave Dittrich und stützen uns auf das Beispiel aus Listing 3.

Wir können ihm ähnliche Grunddaten wie früher dem Programm `capinfos` entnehmen. Diesmal stehen uns allerdings sogar mehr Informationen zur Verfügung, und zwar die Anzahl der Pakete in den einzelnen Größenbereichen sowie detailiertere Angaben zu Verkehrsprotokollen (die Informationen sind unter *Protocol Breakdown* zusammengefasst). Die einzelnen Zeilen folgen dem Format:

```
[Protokoll-"Ebene"] Protokoll Paketzahl
 (prozentuell_im_Bezug_auf_alle_Pakete)
 Bytezahl (prozentuell_im_Bezug_
auf_den_Gesamtumfang) durchschnittlich_
Bytes_pro_Paket
```

Dank diesen Angaben können wir ermitteln, was für Pakete der einzelnen Protokollfamilien in unserem Netzwerk übertragen werden. Dies ist zu beachten, weil wir anhand der hier aufgelisteten Protokolle feststellen können, ob unser Netzwerk nicht fernanalysiert oder gar angegriffen wird.

Zu beachten ist die Häufung von TCP-Paketen. Während Informationen über HTTP-Pakete uns nicht zu beunruhigen brauchen (solange es keine bösartige Nutzung unserer PHP-Anwendungen ist), sollten wir uns überlegen, was für den intensiven übrigen TCP-Verkehr (other) verantwortlich ist. Dies ist der Verkehr, der in pcap anhand der Vermittlungsschicht, allerdings ohne Portangabe definiert ist. Ty-

Listing 3. *Ausgabe des tcpdstat-Programms*

```
DumpFile: traffic.cap
FileSize: 0.09MB
Id: 200506231455
StartTime: Thu Jun 23 14:55:18 2005
EndTime: Thu Jun 23 15:01:01 2005
TotalTime: 342.14 seconds
TotalCapSize: 0.07MB CapLen: 68 bytes
# of packets: 1194 (208.31KB)
AvgRate: 5.08Kbps stddev:30.22K
### IP flow (unique src/dst pair) Information ###
# of flows: 66 (avg. 18.09 pkts/flow)
Top 10 big flow size (bytes/total in %):
20.0% 16.3% 15.7% 12.9% 4.8% 4.0% 2.9% 1.3% 1.3% 1.2%
### IP address Information ###
# of IPv4 addresses: 68
Top 10 bandwidth usage (bytes/total in %):
69.9% 21.5% 18.5% 17.5% 16.9% 13.9% 5.4% 5.2% 4.5% 4.3%
# of IPv6 addresses: 4
Top 10 bandwidth usage (bytes/total in %):
81.5% 59.2% 40.8% 18.5%
### Packet Size Distribution (including MAC headers) ###
<<<<
[ 32- 63]: 857
[ 64- 127]: 104
[ 128- 255]: 79
[ 256- 511]: 61
[ 512- 1023]: 14
[ 1024- 2047]: 79
### Protocol Breakdown ###
<<<<
protocol packets bytes bytes/pkt
-----------------------------------------------
[0] total 1194 (100.00%) 213308 (100.00%) 178.65
[1]  ip 988 ( 82.75%) 198381 ( 93.00%) 200.79
[2]   tcp 884 ( 74.04%) 180408 ( 84.58%) 204.08
[3]    http(s) 219 ( 18.34%) 124825 ( 58.52%) 569.98
[3]    other 665 ( 55.70%) 55583 ( 26.06%) 83.58
[2]   udp 94 ( 7.87%) 17247 ( 8.09%) 183.48
[3]    dns 9 ( 0.75%) 2752 ( 1.29%) 305.78
[3]    other 85 ( 7.12%) 14495 ( 6.80%) 170.53
[2]  icmp 7 ( 0.59%) 546 ( 0.26%) 78.00
[2]  igmp 3 ( 0.25%) 180 ( 0.08%) 60.00
[1] ip6 5 ( 0.42%) 422 ( 0.20%) 84.40
[2]  icmp6 5 ( 0.42%) 422 ( 0.20%) 84.40
```

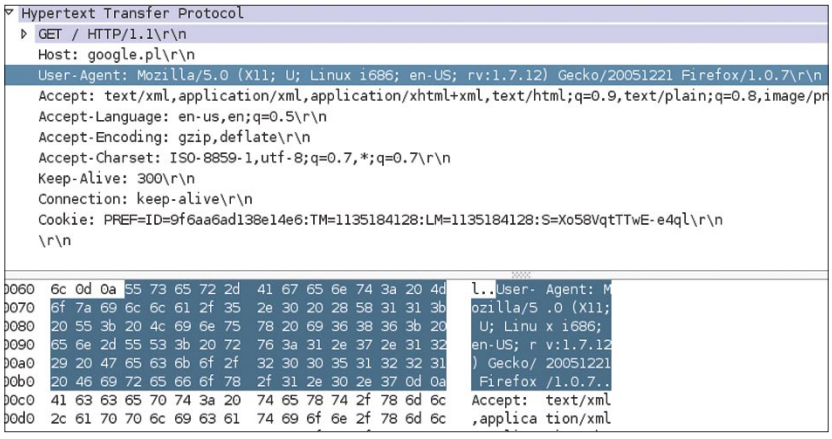


Abbildung 1. *Download von Webdokumenten-Beispielpaket*

n.runs...

network · technology · security · consulting

... Knowledge makes the Net Work

Die *n.runs GmbH* mit Sitz in Oberursel bei Frankfurt bietet Großunternehmen und Service Providern herstellerunabhängiges IT-Security Consulting für alle Bereiche von der Strategie, über Policies bis hin zu Penetrationstests, Quellcode-Reviews und forensischen Analysen. Die Projekte umfassen Strategieberatung, Planung und Design, Implementierung und insbesondere Sicherheitsanalysen.

Wir wollen weiter wachsen. Daher suchen wir (bundesweit) mehrere

PENETRATION TESTER

Die Hauptaufgabe dieses Security Consultants besteht darin, unter Verwendung der n.runs Methodologien hochqualitative Sicherheitsüberprüfungen in komplexen und kritischen Infrastrukturen zu bieten, und adäquate Lösungsstrategien zu erarbeiten. Diese Beratung wird zumeist als Teil eines Teams geliefert.

Wir erwarten eine mindestens 3-jährige Berufserfahrung im Bereich Information Security, vorzugsweise in der Sicherheitsberatung. Die Fähigkeit zu einer kompetenten Beratung setzen wir ebenso voraus wie den Nachweis, dass keine Vorstrafen vorliegen.

Umfassende Kenntnisse und Erfahrung in den folgenden Schwerpunkten sind Voraussetzung

- Durchführung eigenständiger, händischer Penetrationstests
- Penetrationstest Spezialthemen wie Wardialing, Telefonanlagensicherheit, WLAN Sicherheit, etc.
- Programmierkenntnisse in C und/oder Perl/Python/Ruby
- Quellcode Analysen
- Computer Forensik (gerichtsverwertbar)
- Sehr gute Betriebssystemkenntnisse Unix und Windows
- Reverse Engineering Kenntnisse von Vorteil

Weiterhin setzen wir bei unseren Consultants voraus

- Sehr gutes Deutsch und Englisch in Wort und Schrift
- Fähigkeit, eigenständig und im Team zu arbeiten
- Zuverlässigkeit und Diskretion
- Proaktive und kundenorientierte Arbeitsweise
- Bereitschaft zu Reisen im In- und Ausland
- Starkes Interesse an einer ständigen Weiterbildung
- Ausgeprägtes analytisches Denkvermögen
- Fähigkeit zur Erstellung aussagekräftiger technischer Dokumentationen

Wenn Ihnen neben den wechselnden fachlichen Herausforderungen in Projekten ein kollegiales Team in einem Umfeld, in dem Sie sich entfalten und das Sie mitformen können, mit schnellen Entscheidungswegen, ständiger Weiterbildung sowie leistungsorientiertem Gehalt plus Firmenwagen wichtig sind, dann freuen wir uns auf Ihre Bewerbung (auf Deutsch oder Englisch) unter *pentester@nruns.com*.

weitere Informationen
Telefon (Frau Anke Wittmer): 01 73 - 5 65 79 74
Internet: www.nruns.com

Listing 4. *Ein Skript zum Aufsuchen von IP-Adressen in der pcap-Datei (searchip.pl)*

```
#!/usr/bin/perl

use Switch;
use Net::Pcap;

my $err;
my $rep;
my $curr_ip;
my @ip_table = ();

sub connread;

if ($ARGV[0] eq "")
{
print "Usage: ./search_ip <filename/filepath>";
exit;
}

if($pcap = Net::Pcap::open_offline($ARGV[0], \$err))
# Datei öffnen
{
Net::Pcap::loop($pcap, -1, \&connread, '');
# jedes Paket in die connread-Funktion einspeisen
}
else
{
print "Error\n";
exit;
}

print "Found different IP:\n";
foreach $ip_table(@ip_table)
{
print $ip_table."\n";
}

sub connread
# Hauptfunktion
{
my($data, $header, $packet) = @_;
my $packet = unpack('H*', $packet);
$rep=0;
# IP-Adresse im Paket aufsuchen
if ($packet =~ m/^\w{44}(..)(..)(\w{4}|\w{8})(..)(..)(..)(..)\
                    w{8}(....)(....)\w+(.*)/)
{
# in einer Variable speichern und mit den bereits
# gefundenen vergleichen
$curr_ip = hex($4).".".hex($5).".".hex($6).".".hex($7);
foreach $ip_table(@ip_table)
{
if($ip_table eq $curr_ip)
{
$rep = 1;
}
}
if ($rep == 0) {
# wenn die IP-Adresse noch nicht aufgezeichnet wurde,
# ins Array eintragen
push(@ip_table, $curr_ip);
}
}
}
```

pischerweise ist es ausgehender Verkehr, aber nicht immer – es kann auch ein Versuch sein, unser System zu scannen.

Eine weitere Information, die *tcpdstat* liefert, sind u.a. die 10 höchsten Netzwerkauslastungsfälle. Derartige Daten brauchen wir allerdings nur für Statistiken.

Jetzt, wo wir uns über Pakete und Protokolle informiert haben, wären detailierte Angaben über die Empfänger und Sender von Paketen nützlich. Dafür bedienen wir uns des einfachen Skripts aus Listing 4.

Die Ausgabe des Skripts listet IP-Adressen der Pakete auf, die über unseren Host übertragen werden (dabei sind es nur Quellpakete). Schon anhand dieser Informationen können wir Firewallregeln aufstellen, die die Kommunikation mit den einzelnen IP-Adressen ablehnen bzw. zulassen. Zu prüfen ist auch, ob sich die Adressen nicht im *bogon space* befinden.

Ist es der Fall, dann bedeutet das, dass in unserem Netzwerk ein erfolgloser DDoS-Angriff durchgeführt wird bzw. wurde. Dann wäre (für ein paar Tage) ein Skript nützlich, das Bogon-Adressen an der Firewall sperrt (solch ein Skript steht unter *http://completewhois.com/* zum Download bereit).

Eine kleine Modifizierung von `searchip` liefert die MAC-Adressen der Interfaces, von denen die untersuchten Pakete stammen. Diese brauchen wir hier allerdings nicht, weil wir den eingehenden Verkehr aus dem Internet, und nicht aus dem Lokalnetzwerk analysieren.

Bogon Space

Kurz gesagt, ist *bogon space* ein Internetadressraum, der noch keinen Besitzer hat. Pakete, die von solchen Adressen stammen, sollten als ungültig klassifiziert und verworfen werden. Eine laufend aktualisierte Liste von Bogons wird unter *http://completewhois.com/* geführt.

Mehr über *bogon space* und *bogon packets* lesen Sie in hakin9 Nr. 5/2005 nach.

Was DoS- und DDoS-Angriffe angeht, sind sie solch ein weites Feld, dass ein separater Artikel darüber notwendig wäre. In hakin9 Nr. 5/2004 haben Andrzej Nowak und Tomasz Potęga beschrieben, wie man sich gegen laufende Angriffe wehren kann.

Dabei hat das Sperren von DDoS-Angriffen, die bereits statt gefunden haben, wenig Sinn, da der bzw. die Angreifer bestimmt einen anderen Adressenpool verwendet bzw. verwenden.

HTTP, FTP – Analyse der Informationen

Fragen wir uns nun: Sind wir im Stande, einer detailierten Analyse unseres Webservers und ihren Folgen vorzubeugen? Können wir unseren FTP-Server vor der Ausnutzung zum Portscannen schützen? Die Antwort ist in beiden Fällen *ja*, es ist allerdings keine einfache Aufgabe. Manuell durchgeführt, ist sie enorm zeitaufwändig. Überzeugen wir uns darüber an einem kleinen Beispielausschnitt.

Wir öffnen die Verkehrsaufzeichnungsdatei mit dem *ethereal*-Tool (das wir vor allem für seine grafische Oberfläche einsetzen). Wir wählen eins der Pakete, die den Download von Webdokumenten starten (das Paketpayload muss das Wort GET bzw. HEAD enthalten).

Solch ein Beispielpaket steht in Abbildung 1. Offenbar enthält jede *gute* Verbindung Informationen, die automatisch durch den Browser angegeben werden, u.a.: den Typ und die Version des Browsers (User-Agent), den akzeptierten Dateityp (Accept), die akzeptierte Zeichencodierung (Accept-Encoding), die bevorzugte Sprache (Accept-Language). Außerdem können Anweisungen zum Aufrechterhalten der Verbindung (ob und für wie lange) sowie eine Cookie-ID auftreten.

Praktisch alle Netzeindringlinge lassen beim Untersuchen der vom Server freigegebenen Informationen (seiner Banner) Browserangaben aus (weil Server die länger inaktiven

Listing 5. *fhelp.pl*

```
#!/usr/bin/perl
use Switch;
use Net::Pcap;

my $dev = "eth0"; # das belauschte Interface
my $pcap;
my $err;
my $packet;

sub connread;

if ($ARGV[0] eq "-h") {
print "Usage: ./fhelp.pl <filename>\n\r If no filename given will start live
                        capture\n";
exit;
}
# überprüfen, ob ein Dateiname vorgegeben wurde, wenn ja, öffnen,
# sonst "live capture" öffnen
if ($ARGV[0] == "") {
if($pcap = Net::Pcap::open_live($dev, 2000, 1, 1000, \$err)) {
Net::Pcap::loop($pcap, -1, \&connread, '');
}
else {
print "Error\n$err\n";
exit;
}}
else {
if($pcap = Net::Pcap::open_offline($ARGV[0], \$err)) {
Net::Pcap::loop($pcap, -1, \&connread, '');
}
else {
print "Error\n$err\n";
exit;
}}


sub connread {
my ($data, $header, $packet) = @_; $get = "n";
my $packet = unpack('H*', $packet); # Paket auslesen und entpacken
# Verbindungen zu den Ports 80 und 21 suchen
if ($packet =~ m/^\w{44}(..)(06)(\w{4}|\w{8})(..)(..)(..)(..)\w{8}(....)(005
                        0|0015)(.*)/) {
$curr_ip = sprintf("%d.%d.%d.%d", hex($4), hex($5), hex($6), hex($7));
$traffic = sprintf("%s",$10);

# inkomplette HTTP-Pakete suchen
if ($traffic =~ /(474554|48454144)/) {
if (!($traffic =~ m/^(.*)\w(20485454502f312e(31|30)0d0a)(.*)(557365722d416765
                        6e743a)(.*)/)) {
print "incomplete http header from $curr_ip\n";
print "do you want to block ip $curr_ip on iptables? [y/N]: ";
$get = <>;
if (($get eq "y")|($get eq "Y")) {
system("iptables -A INPUT -p tcp -s $curr_ip -j DROP");
}
}}
# port-Befehle in FTP-Paketen suchen
if ($traffic =~ /(706f7274|504f5254)/) {
print "PORT command used in ftp connection from $curr_ip\n";
print "do you want to block ip $curr_ip on iptables? [y/N]: ";
$get = <>;
if (($get eq "y")|($get eq "Y")) {
system("iptables -A INPUT -p tcp -s $curr_ip -j DROP");
}}
}}
```

Verbindungen abbrechen). Diese Tatsache können wir uns zu Nutze machen.

Wenn wir beim Durchsuchen der Verkehrsdaten inkomplette Header vorfinden, können wir vermuten, dass unser Server analysiert wurde, und uns auf einen kommenden Angriff einstellen. Leider, oder vielleicht zum Glück, ist es nicht immer wahr, weil textbasierte Browser (wie *Lynx*, *links*) keine solch "Karte" hinterlassen. Eine Verwechslung ist hier also möglich, allerdings stellen diese Anwendungen Cookieinformationen ein, was sie von den serveranalysierenden Angreifern unterscheidet.

Sehen wir uns jetzt das FTP an. Wenn wir einen anonymen FTP-Server bereitstellen, kann er zum Scannen der Ports eines beliebigen Servers ausgenutzt werden, und zwar mit dem Befehl PORT. Er kann natürlich gesperrt werden, damit deaktivieren wir aber auch aktive Verbindungen zu unserem Server. Wenn wir sie aus irgendeinem Grund doch brauchen, wäre eine FTP-Verkehrsanalyse auf unserem Server ab und zu empfehlenswert. Wie bei HTTP, könnten wir stundenlang nach einzelnen IP-Adressen suchen und sie an der Firewall sperren, es geht hier aber um Effizienz. Abhilfe schafft das Skript *fhelp.pl* (siehe Listing 5).

Das einfache Skript durchsucht die vorgegebene Datei im pcap-Format bzw. arbeitet *live*, indem es fehlende Browserangaben in HTTP-Paketen aufsucht. Es analysiert auch den FTP-Verkehr und sucht darin nach dem PORT-Befehl. Wird dieser irgendwo vorgefunden, fragt das Programm den Benutzer, ob die entsprechende IP-Adresse an der Firewall (nur *iptables*) gesperrt werden soll. Natürlich müssen wir das Skript als Root starten.

SSL – die Achillesferse

Jeder, der sich mit Netzwerkanalysen befasst, begegnet früher oder später verschlüsseltem Verkehr. Es ist ein klassisches Problem der meisten Analysesitzungen.

Diesen Verkehr zu entschlüsseln ist zwar möglich, wie die Praxis aber zeigt, völlig unökonomisch. Erstens ist es in 999 von 1000 Fällen legaler Verkehr. Zweitens kostet die Entschlüsselung der Pakete unproportionell viel Zeit, weil der Sequenzschlüssel für jede Übertragung ermittelt werden müsste. Wenn man sich solche Übertragungen überlegt, gelangt man schnell zu dem Schluss, dass die einzig sinnvolle Lösung ist, nur vertrauten Hosts verschlüsselte Verbindungen zu erlauben. Leider bewährt sich dieser Ansatz nicht, wenn unser Server öffentlich zugänglich sein soll und verschlüsselte Verbindungen mit den Clients der angebotenen Dienste voraussetzt.

Unehrliche Mitarbeiter

Hier wäre das Perlskript *zonk.pl* (vom Autor dieses Artikels) zu erwähnen.

Diese einfache, auf regulären Ausdrücken und der *pcap*-Bibliothek basierte Anwendung sucht im laufenden Netzwerkverkehr nach Symptomen einfacher Leitungsüberlastung (durch *P2P-*, *IM-*, *IRC*-Software), und zwar den durch die *verbotenen* Dienste Serverportnummern nach. Das Skript kann sich für Administratoren von Unternehmensnetzen als nützlich erweisen, indem es eine übermäßige und unerlaubte Nutzung von Firmenressourcen durch Mitarbeiter feststellen und unterbinden lässt. Zonk steht, wie auch die anderen im Artikel angesprochenen Skripte, auf der Website des Autors (siehe Kasten *Im Internet*) zum Download bereit. Auch weitere administrative Tools sind dort zu finden.

Wozu das alles?

Wenn man die oben angesprochenen Einschränkungen bedenkt, stellt man sich vielleicht die Frage, ob sich die Verkehrsanalyse in einem Netzwerk überhaupt lohnt.

Sie ist ganz sicher kein Wundermittel gegen alle unsere Probleme, die in Folge einer korrekt und zur rechten Zeit durchgeführten Analyse angesammelten Informationen können sich jedoch als unschätzbar erweisen. So können *brute force*, DoS- und DDoS-Angriffe, P2P- und IM-Verkehr sowie übermäßige Auslastung von HTTP-Verbindungen festgestellt werden.

Natürlich ist der *rechte Zeitpunkt* für die Analyse besonders schwer festzulegen. Daher lohnt es sich, sie regelmäßig durchzuführen und Skripte einzusetzen, die größeren Anomalien automatisch erkennen. Wir müssen auch daran denken, dass das wichtigste Element der durchgeführten Analyse nicht die Ansammlung von Daten, sondern eine Interpretation des abgefangenen Verkehrs ist.

Außerdem müssen wir damit rechnen, den ganzen Verkehr nach der Verwendung aller Tools und Skripte noch persönlich untersuchen zu müssen, um kleinere Anomalien zu erkennen. Dies ist aber eine Folge, deren sich jeder Netzwerkanalytiker bewusst sein muss. ●

Im Internet

- *http://www.ethereal.com/* – die Homepage des Sniffers ethereal,
- *http://www.tcpdump.org/* – die Homepage von libpcap und tcpdump,
- *http://www.netfilter.org/* – die Homepage des iptables-Projekts,
- *ftp://tracer.csl.sony.co.jp/pub/mawi/tools/* – von hier kann tcpdstat downgeloadet werden,
- *http://aqu.banda.pl/scripts/network_analyzing/* – von hier kann zonk.pl downgeloadet werden.
- *http://www.skyfillers.net/Iris.iris-netzwerk.0.html* – Iris ist ein Network Traffic Analyzer zur proaktiven Überwachung des Netzwerkes;
- *http://www.microsoft.com/germany/technet/datenbank/articles/600573.mspx* – Testen von Netzwerkpfaden für die am häufigsten verwendeten Arten von Netzwerkverkehr