Linux Security

am 19.Feb.2001

Von Benjamin Heitmann <benjamin@LauschAngriff.org>

www.ulm.ccc.de
www.LauschAngriff.org

Der Vortragende ist nicht perfekt, deswegen kleinere Thinkos und Typos bitte sofort melden, und nach dem Vortrag ist Zeit fuer Nachfragen und Kommentare

Zielsetzung des Vortrags

- dieser Vortrag ist sehr weit gestreut,
- er wird eher selten in die Tiefe gehen
- es soll eher probiert werden das "Mindset" welches noetig ist um Linux Security zu verstehen, zu ueberliefern. (Wer dieses verstanden hat wird sofort sehen das man sowieso viel Zeit in das Thema stecken muss um Sicherheit erreichen)
- der Vortrag will eher die richtigen Pointer in die Bereiche geben, in denen jeder sich selber weiterbilden sollte
- "Sicherheit ist kein Produkt, sondern ein Prozess"
  Bruce Schneier, aus „Secrets and Lies“

Wieso ist Linux Security wichtig ?

- Security ist immer wichtig wenn man
  - wichtige Daten auf dem Rechner hat
  - Dienste fuer andere anbietet
  - überhaupt einen Rechner in einem Netzwerk mit anderen Rechnern betreibt
- Wer im Netz hängt und nicht auf die Sicherheit seines Rechners achtet, handelt eigentlich in gewissem Masse fahrlässig gegenüber dem Rest des Internets. (Diese Reaktion war etwa nach den DDoS Attacken Anfang 2000 oft zu hören.)
- Ein wirklich sicherer Rechner hängt an einer autonomen Stromversorgung in einem Bunker unter der Erde in einem Tresor ohne Netzwerk
- Linux Server sind heutzutage nicht von Haus aus sicher sondern, genauso wie Windows NT/2K Server erst „abzudichten“

Der Ramen Wurm

- der Ramen Wurm ist der jüngste Beweis für die Faulheit und Unkenntnis einer grossen Menge von Linux „Systemadministratoren". Er wurde um den 18.Jan.2001 in freier Wildbahn gesichtet.
- es wird nach RedHat Linux 6.2/7.0 gescannt (im FTP Banner erkenntlich)
- er benutzte Löcher in den Paketen:
  - wu- ftpd (fuer RedHat 6.2, Fix vom 23.6.2000, >6 Monate)
  - statdx Exploit im nfsd der nfs- utils (RH 6.2, Fix vom 17.7.2001, >6 Monate)
  - syslog format bug im LPRng (RH 7.0, Fix vom 26.9.2000, >3 Monate)
- falls ein Angriff erfolgreich war, wird von der infizierenden Maschine ein File per HTTP gehohlt und auf der infizierten Maschine ein Mini- HTTP Server auf Port 27374 gestartet, eine Mail an Hotmail und Yahoo (gb31337@hotmail.com) verschickt, sowie nach weiteren Zielen gescannt
- der Wurm ist aus verschiedenen Exploits zusammengesetzt worden (nicht alle Binaries sind gestripped worden) und attackiert Standarddienste auf veralteten Linux Rechnern

Der Ramen Wurm

- alle auf dem infizierten Rechner gefunden HTML Dateien werden mit dem Text "RameN Crew - - Hackers looooooooooooove noodles." ersetzt.
- es wurden leider beim Scannen nach potentiellen neuen Zielen auch Multicast Adressen gescannt. Und das sich dadurch einige Leute angepinkelt gefühlt haben, dürfte klar sein. (Es gibt z.Z. AFAIC noch kein Sicherheitssystem bei Multicast Anwendungen)
- bei aufgespielten Updates hatte der Wurm keine Chance. Zum Aufspielen war viel Zeit vorhanden
- mit leichten Abwandlungen wäre der Wurm nicht Distributionsspezifisch gewesen
- zur Zeit laufen laut NetCraft.com 70% aller Linux WebServer mit einer RedHat Version
- es gibt keine genauen Zahlen wie viele Rechner infiziert wurden, aber unter den infizierten Sites waren die NASA, das Jet Propulsion Lab in California, ein University of Texas A&M server, and ein Supermicro Server in Taiwan.

Themenbereiche für Heute sind:

1.) Basic/Fundamental Linux Security
(das sollte jeder wissen)

2.) Advanced Linux Security
(das sollte jeder lernen)

3.) Esoteric/Far–Out Linux Security
(wer das weiss ist dem Rest einen Schritt vorraus – Ist aber nicht arg viel :(

1.) Basic/Fundamental Linux Security

•

Angriffsarten und Motivationen

- wichtigsten Motivationen:
  - Sabotage (Ausfall eines DNS Servers führt zur Unnerreichbarkeit einer Webseite)
  - Informationsgewin (Welche Geschäftspläne hat Firma XYZ)
  - unerlaubte Verwendung von Diensten (Durchführung von Kreditkartentransaktionen bei Online Shops zu Gunsten des Angreifers)
  - Vorteile gegenüber Dritten Parteien (Verwendung des Ziels als Plattform für Angriffe gegen das primäre Ziel)
- wichtigste Durchführungsziele:
  - DoS (Denial of Service)
  - Übernahme des Rechners durch Erwerb von Systemadministrator/Root Privilegien

Angriffsarten und Motivationen

- wichtigste Methoden um das Ziel anzugreifen:
  - physikalischer Zugriff vor Ort. Wird meistens vergessen. Daten auf einer ausgebauten Festplatte sind nicht mehr wirklich sicher, oder „init=/bin/sh“ als LILO Parameter ist oft nicht verboten
  - durch Programme, die auf der Netzwerkkarte nach Daten lauschen (Bufferoverflows. Besonders peinlich bei Sniffern)
  - durch Programme, die es Benutzern erlauben sich einzuloggen (Sniffen bei unverschlüsselten Diensten, Man in the Middle Attacke bei verschlüsselten Diensten)
  - als lokaler Benutzer, der ein harmloses Program in einer nicht geplannten Vorhergesehenenweise verwendet (tmp−races, Bufferoverflows. Auch ein PacMan Clone kann so missbraucht werden)

Pflichten der Wurzeln

- nur Software installieren die auch benötigt wird, auch wenn es schwerfällt im Zeitalter der 30 GB IDE Platten nicht einfach auf "Alles" zu klicken und sich den späteren Nachinstallierärger zu sparen
- nur Dienste starten die auch benötigt und verwendet werden.
  Merke: lässt sich die Funktion und die Abhängigkeit von diesem Dienst nicht in 2 Sätzen kurz umschreiben, dann runter damit. Überzeugen das auf jedem Port auch der richtige Dienst läuft.
- immer und überall auf Verschlüsselung achten:
  - verschlüsselten Diensten sind gegenüber unverschlüsselten gleichartigen Diensten der Vorzug zu geben, auch bei erhöhtem Administrationsaufwand
  - bei der Installation eines Systems auf MD5 Passwortdateien achten
  - eventuell Partitionen verschlüsseln

Pflichten der Wurzeln

- Problem der Standarddienste: BIND, Sendmail und Apache sind fast auf jedem System vorhanden, da sie ihre Aufgabe optimal erfüllen.
- Die Suche nach sicheren Alternativen ist trotzdem meistens sinnvoll, wenn auch nicht unbedingt zufriedenstellend.
- Es gilt hier jedoch leider eine Regel der Biologie:
  Monokulturen begünstigen Schädlinge
- Vertrautmachen mit den wichtigsten Log–Files. (unter /var/log)
  Regelmässig selbst nach Anomalien schauen. Früher oder später wird einem dieses Wissen helfen.
- Problem der unverschlüsselten Kommunikation an sich: wenn Dienste bei denen sich der Benutzer des Dienstes authetifizieren muss, unverschlüsselt oder ohne Digest Authentifzierung ablaufen, oder Daten unverschlüsselt übertragen werden, besteht immer die Moeglichkeit, das Informationen an unerwünschte dritte Parteien durchsickert.

unverschlüsselte Dienste

- Telnet : klassisches Beispiel, viele wissen gar nicht das das unervschlüsselt ist und nehmen an das "da schon irgendwie Sicherheit drin ist"
- FTP: noch schlimmer. wird von so gut wie jedem benutzt, z.B. bei den ganzen billig WebSpace Aldis. Dabei kann das ganze Internet das Passwort mitsniffen.
- SMTP: Emails werden komplett unverschuesselt und unauthentifiziert übertragen
- POP3: auch alles unverschluesselt
- ICQ: so gut wie alle Instant Messaging Dienste werden unverschluesselt uebertragen, obwohl es auch anders gehen wuerde

verschlüsselte Dienste

- SSH statt Telnet: alles verschluesselt, auch der Handshake. Hat sich mittlerweile etabliert
- Alle obigen Dienste lassen sich unter mehr oder weniger grossem Aufwand mit SSL nachruesten
- PGP/GPG: um Emails zu verschluesseln. Ist bekannt aber nicht etabliert

Distributionsspezifische Aufgaben

- Verstehe deine Distribution. Zielsetzung, Marktposition, interne Mechanismen
- auf den jeweiligen Security Mailinglisten subscribed sein
- schnell und rechtzeitig Updates einspielen
- Überblick der Update–Mechanismen der wichtigsten 3 Distributionen
  - SuSE Linux. Dominant in Europa. Kein automatisches, vom Hersteller unterstütztes System. Mailingliste vorhanden. Responsezeit gut bis ausreichend
  - RedHat Linux. Dominant in Amerika. Automatisches System über das RedHat Network mit Registration und dem grafischen Frontend up2date. Mailingliste vorhanden. Responszeit gut.
  - Debian. Simply rules. Automatisches System mit Frontend oder unattended. Freundliche Mirrors in deiner Nachbarschaft. Seit kurzem auch mit signierten Packeten. Mailingliste vorhanden. Responsezeit sehr gut.

Security News

- aktuelle Informationen zu erhalten ist wichtig. Nichts ist so alt wie ein Adivsory von gestern, nachdem man mit dem darin beschriebenen Bug gecrackt wurde.
- Wo informiert man sich also, und wieviel Zeit kostet das?
  - auf der Security Mailingliste der eigenen Distribution subscribed sein
  - auf Bugtraq subscribed sein wenn Zeitaufwand angemssen erscheint (realistisch sind 10 bis 15 Minuten Pro Tag.)
  - eine NewsSeite pro Tag Besuchen. Zu empfehlen sind: Heise News oder auf Englisch im Linux- Bereich Linux Today. (ca. 5 bis 10 Minten pro Tag)
  - entsprechende Gruppen im Old- School Usenet (ca. 10 bis 20 Minuten, Noise/Signal Ratio schlimmer als auf Bugtraq)
  - Verschiedene monatliche und wöchentliche Newsletter. Diese zeigen Tendenzen an, unregelmaessiges Ueberfliegen reicht deswegen meistens (zB von Bruce Schneier, monatlich Cryptogram. http://www.counterpane.com/crypto- gram.html)

Security News

- Papiermedien: ct und ix in Deutschland. Ansonsten sind die meisten Zeitungen die man so am Kiosk findet eher nicht ernst zu nehmen. (5 Stunden pro Monat)
- Achtung: seit kurzem stellen auch mache der Security Consulting Firmen, welche Advisories herausgeben, Copyrightansprüche auf ihrer Advisories. So geben etwa Microsoft und @Stake (da arbyten jetzt die Jungs von L0pht) nicht alle ihrer Advisories frei.
- Jede Firma oder Organisationseinheit sollte mindestens eine Person haben die sich wenigstens mit einem Teil ihrer Zeit um die Sicherheit kümmert. Dabei sollte jeden Tag eine bestimmte Zeit eingeplannt werden, wobei täglich und kurz besser ist als lang und nur einmal die Woche.
- Wie wichtig es ist regelmässig die Nachrichten im Sicherheitsbereich zu verfolgen und auch schnell zu reagieren zeigte vor kurzem das Bekanntwerden von mehreren gravierenden Sicherheitslücken im BIND.

Das BIND Scheunentor

- Chronologie der Abläufe:
  - Ende Dezember 2000 werden die Löcher im BIND gefunden
  - es werden im Stillen von allen grossen Herstellern Updates gemacht, nachdem die Löcher vom Internet Software Consortium geflickt wurden. Es gibt Exploits, die jedoch nicht der Öffentlichkeit zugänglich waren
  - am Montag, 22.1.2001 wurde ein Advisory dazu auf Bugtraq und anderen relevanten Listen gepostet
  - am Dienstag stand davon auf Slashdot und auf Linux Today, nachmittags dann auch auf Heise. Debian Packete ohne Loch sind fertig.
  - am Mittwoch sind gefixte Packete von RedHat und SuSE da. Ein Fake Exploit taucht im Uppergroundauf, und vermutlich taucht auch ein echter Exploit im Underground auf.
  - am Donnerstag ist die Ärger im Anmarsch. Wer jetzt noch nicht seine Bindyleins ge- updated hat, merkt es spätestens jetzt wenn sein Server spinnt.

Das BIND Scheunentor

- Die detailierte Tabelle der Bind Löcher befindet sich auch jetzt noch unter: http://www.isc.org/products/BIND/bind-security.html
- Die Bind Versionen bei Aktuellen Distrubtionen waren alle mit ca. 5 Löchern versehen, davon 3 mit Möglichkeiten zum Ausführen von Code als Root, sowie mit Möglickeiten zum entfernten (remote) auslösen.
- Zeitfenster des Otto-Normal-Admins nachdem er vom Bug erfahren hatte bis zu den ersten im grossen Stil angewendeten Exploits war von Dienstag bis Donnerstag morgen. Also ca. 3 Tage.

2.) Advanced Linux Security

Linux Systemarchitektur spezifisches

- Attacken auf Löcher im Kern selber.
  Gab es bei jeder Major Version des Kerns. Manchmal wurde im Nachhinein festgestellt das die Bugs fuer Monate drin waren. Besonders boeses Beispiel: the infamous "ping me tender"- Bug im 2.0er Kern. Ein simpler Ping mit einer Paketgroesse von 65468 konnte eine Kiste einfrieren (ping - s 65468 deadmeat.victim.org) , und der Bug wurde im 2.0.38 Kern gefunden und erst im 2.0.39 gefixt. (Okay, einen Patch gab es schon vorher.) Doch auch im 2.4 gab es schon einen Sysctl Bug der duch Angabe eines negativen Byte- Offsets das auslesen von grossen Bereichen des Speichers erlaubte.

- PAM (Pluggable Authentication Modules) von Sun erfunden, aber seit 2 Jahren bei allen Linux Distributionen dabei. Damit wird etwa möglich:

  - andere Passwortverschlüsselungen (zB MD5 oder triple ROT- 26 ;)
  - Authentifzierung eines Linux Rechners gegen eine NIS/NIS+/Windows Domain
  - Resourcenlimits für Benutzer setzen und Restrictionen fuer die Zeit des Einloggens setzten

Linux Systemarchitektur spezifisches

- glibc – Die wunderbare neue Welt der neuen GNU Lib C ab Version 2.0 bringt so spassige Sachen wie mehrere Versionen der Libc in einem File mit. Leider auch einen Haufen nicht ganz ausgereifter Interfaces. Speziell bedauerlich waren neulich 2 Bugs in defensiven Features, welche eigentlich vor Missbrauch schützen sollten.
  - ein Bug im Handling von von LD_PRELOAD und LD_LIBRARY_PATH Shell Umgebungsvariablen bei set_uid Programmen. Leider hat es manchmal damit gehackt
  - eine Funktion zur Überprügung der LANG Variablen machte genau das Gegenteil, sie schloss keine Lücken sondern öffnete neue Lücken

Linux Systemarchitektur spezifisches

- normale Dateiattribute das ext2 Dateisystems
  - das „sticky bit“ erlaubt bei einem Verzeichniss nur dem Besitzer einer Datei oder einem Benutzer mit expliziter Schreibberechtigung diese zu Löschen auch wenn dem Benutzer das ganze Verzeichniss gehört
  - ausserdem Vorsicht mit dem Set_uid Bit und dem Set_gid Bit. Damit bekommen ausführbare Datein immer die Rechte ihrer Besitzer.
- erweiterte Dateiattribute des ext2 Dateisystems (ziemlich unbekannt)
  - mit chattr und lsattr zu verändern
  - Attribute sind: append–only flag, online compression (nicht unterstützt), immutable flag, s – beim Löschen wird die Datei mit Nullen ueberschrieben, sync flag, allow undelete flag.
- die neuen Journaling Dateisysteme haben andere Besonderheiten:
  - ReiserFS kann ACLs (Access Control Lists) realisieren
  - SGI's XFS kann angeblich auch viele nette Sachen

Linux Systemarchitektur spezifisches

- RPC (Remote Procedure Call) und NFS (Network Filesystem) eröffneten jeweils für sich viele Bugs in der Vergangenheit
- notwendige Dienste und der TCP Wrapper.
  Es sollten wie schon gesagt nur die nötigsten Dienste auf einer Kiste laufen, und diese sollten weitestgehend konfiguriert werden um nur in einer klar definierten Weise benutzt zu werden. Falls ein Dienst diese Möglichkeit nicht selber bietet, sollte er mit TCP Wrapper Support kompiliert werden. Dann kann mit den Dateien /etc/hosts.allow und /etc/hosts.deny der Zugriff kontrolliert werden. Man sollte sich auch gleich davon überzeugen das die DNS Informationen über die eigene Domain richtig sind, da sonst Spoofing möglich ist.
- identd. Dieser Daemon gibt anderen System Auskünfte über locale Benutzer. Es sollte eine sichere Variante von ihm installiert sein, und er darf nicht zu Plaudertaschig konfiguriert sein.

Linux Systemarchitektur spezifisches

- Toll und neu: CORBA (Common Object Request Broker Architecture)
  Junge und aufstrebende Desktop Oberflaechen wie KDE und GNOME streben alle nach dem Heiligen Gral der Componentenbasiertheit. Wir alle Hoffen das dabei die Sicherheit nicht geopfert wird. Bisher wurde jedoch etwa von Ximian ganz explizit geäussert, das Sicherheit keine ihrer Top- Priotitäten ist. Und ein unsicheres Netzwerk- Componenten Model hätte noch fatalere Folgen als das tolle M$ Active X. Der Gnome ORBit und das KDE OpenParts Model zeigen jedenfalls noch nicht viele Sicherheitsfeatures.

- Linux Hardening Script

  - fuer RedHat: http://www.bastille- linux.org/
    Das Script begleitet einem beim Zurechtsetzen der sicherheitsrelevanten Eigenschaften des Systems

  - fuer SuSE: harden_suse

- Linux Intrusion Detection System Patch fuer den Kern: http://www.lids.org
  Kann Prozesse und Files unsichtbar machen. Keine neuen Module, Filesysteme oder Konfigurationsdateien nach dem versiegeln des Systems. Root hat nicht mehr unbegrenzte Rechte.
  Sehr unorthodoxe Features, dafür aber sehr pragmatisch.

Intrusion Detection

- Der Befehl „strings <Datei>“ zeigt Zeichenketten in einem Executable an
- mit „md5sum <Datei>“ kann man sich selbst kurz die Prüfsumme anzeigen
- Dateiintegritäts Überprüfungsprogramme (integrity checker)
  Überwachen Änderungen an Dateien und benachrichtigen den Admin danach.
  - Dazu hat sich wohl Tripwire bewährt, auf www.tripwire.com. Ist inzwischen GPL.
  - Ansonsten die Ersatzprojecte AIDE oder Osiris.
  - auf einem RPM basierten System ist ansonsten der simpelste Check das Ausführen von „rpm - Va“.
- Intrusion Detection auf der Basis von Netzwerkverkehranalyse mit Snort:
  anhand von Angriffssignaturen wird der Netzwerkverkehr auf Angriffe überprüft
- Portsentry ist ein Daemon welcher bei versuchten Portscan- Angriffen Alarm schlägt

sonstige Tools

- beliebte Packetsniffer:
  - etherreal: gibt es als GUI und als ncurses Version. Viele Formate von Packet–Log–Dateien importierbar
  - dsniff: kann viele Protokolle sinnvoll abfangen, unter anderem auch SSH mit einer Man–In–The–Middle Attacke
- mit Syslog–NG lassen sich Logfiles auf andere Rechner weiterleiten. Dadurch kann ein Verändern des Logfiles vermieden werden (ausser der Logserver ist gecrackt)
- The Coroners Toolkit: Falls eingebrochen wurde, ist dies eine Werkzeugsammlung, mit welcher gelöschte Dateiinformationen und ähnliches Wiederhergestellt werden kann
- mit Scanlogd lassen sich ungewöhliche Vorkommnisse im LogFile automatisiert zeigen. Braucht aber viel Fine–Tuning

Bufferoverflows

- Grundlage ist das ein Programm Input einer bestimmten Länge erwartet aber nicht überprüft ob der Input länger ist
- Da unter Linux Daten und Code im gleichen Bereich, dem Stack, abgelegt werden, kann ein zu grosser Input dazu führen das nicht Daten sondern Code üeberschrieben wird.
- Wird dieser ausgeführt kann der Angreifer das Ziel dazu bringen ein anderes Programm das grösser ist als nur ein paar Bytes auszuführen
- Es gibt viele obscure Bufferoverflow Varianten, aber das Grundschema ist immer gleich, und als Nicht–Programmierer muss man nicht mehr wissen
- StackGuard und Co. schuetzen nur begrenzt durch „Wächter–Daten" deren Integrität regelmässig geprüft wird. Es lassen sich wegen den Grundlegen immer gleichbleibenden Eigenschaften des Stacks auch immer Wege um diese Schutzmassnahmen finden

Cryptofilesysteme

- Cryptofilesysteme:
  - CFS (Cryptographic File System) sowie das TCFS (Transparent CFS) sind ausgereift. Dazu wird der international Linux Crypto Patch benoetigt
  - Steganopraphisches Filesystem: mehrere Ebenen der Verschlüsselung. Erlaubt es etwa unter Druck Dateien aus dem Dateisystem zu entschlüsseln, ohne aber gleichzeitig alle Dateien preiszugeben. Ein Beweis von noch vorhanden verschlüsselten Datein ist nicht möglich.
  - Self–Certifying Filesystem: lustige verschlüsselte NFS Variante, welche auch über Low Volume/High Latency Links funktionieren sollte

Denial of Service

- Denial of Service, auf Deutsch „Verweigerung des Dienstes“ ist der zu erreichende Zustand nach dem Überheufen des Zieles mit Daten in einer Menge, welche das Ziel nicht schnell genug abarbeiten kann
- Dies führt dann zu Resource Starvation (Ende der Resourcen) oder zu einem Crash irgendeiner Komponente
- Die DDoS Attacken (Distributed DoS) basieren auf einer dreistufigen Topologie mit einer Schicht von befehlsgebenden Rechnern, einer Schicht von angegreifenden Rechnern und einer Schicht von angegriffenen Rechner. Durch die Aufteilung ist eine massive Paralellisierung und eine schlechtere Rückverfolgbarkeit des Angreifers gewährleistet.

Denial of Service

- Berühmte DDoS Tools waren etwa das Teletubby Flood Network oder Stacheldraht. (alle extra für Linux geschrieben)
- Die DoS Attacke an sich ist schon länger bekannt. Meistens werden Smurf Attacken verwendet, auch als ICMP Broadcast Storm bekannt. Dabei wird ein Paket an eine ICMP Broadcast Adresse eines Netzwerkes geschickt, wobei als Absender die Adresse des Opfers gefälscht wird. Die Rechner wollen nun alle Anwtorten und schicken damit ein vielfaches der ursprünglichen Verkehrsmenge an das Opfer.
- Lösung ist Ingres Filtering und ICMP Broadcast Filtering.
- Inzischen werden auch manchmal simple andere Verfahren verwendet

Linux Firewalls

- der Linux Netzwerk Code hat im Laufe der Zeit eine Evolution durchgemacht.
- Linux 2.0: die Grundlagen sind da. Routing und Masquerading von Unix–Diensten kein Problem. Interface ist ipfwadm. Performance unterhalb des BSD Standards.
- Linux 2.2: Auch exotische Dienste koennen masquiert werden. Stark überarbeitetes Interface in Form von ipchains. Sehr flexibel zu Administrieren und auch grössere Freiheiten bei der Gestaltung der Filterdefinitionen. Immer noch schlechtere Performance als die BSDs.
- Linux 2.4: der neue Code ist super flexibel und super neu. Keiner weiss so genau was man damit alles machen kann :) Die Features reichen jetzt auch an die einer guten Cisco ran. Performance ist nun mindestens so gut wie bei den BSDs.

Linux Firewalls mit 2.4

- der NAT Code ist nun stark aufgeräumt
- es gibt Statefull Inspection etwa fuer FTP Sessions oder um extrem Paranoide Regeln aufzustellen
- diverse Zusatzaufgaben lassen sich durch Module erledigen, welche ein sehr simples API zum Kern verwenden
- Traffic Filtering und QoS sind noch möglicher als jemals zuvor :)
- angeblich soll sich sogar eingehender Verkehr begrenzen lassen
- Rückwärtskompatibilität zu 2.0 und 2.2

Linux Firewalls

- anhand von Protokollart (TCP oder UDP) Portnummer sowie eingehender oder ausgehender IP Addresse, Netz oder Device können Pakete erlaubt, abgewiesen oder an andere Subsysteme des Kerns weitergegeben werden (etwa das Masquerading)
- entweder man schreibt die Regeln alle selber per Hand
- oder man verwendet Hilfswerkzeug:
  - Mason oder Nstreams werden scharfgeschaltet, lernen welche Art von Netzwerkverkehr als Normal angesehen werden kann und schreiben Regeln um nur diesen Verkehr durchzulassen
  - Firewall Regeln Verwaltungsprogramme für GNOME oder KDE können den ganzen Prozess stressfreier gestalten, entbinden einen aber nicht von der Pflicht zu verstehen was man tut

Firewalls allgemein

- nicht vergessen:
  Falls die Sicherheits−, Flexibilitäts− oder Wartungsbedürfnisse von einer Linux−Firewall nicht erfüllt werden können, gibt es immer noch kommerziele Produkte, wie die CheckPoint Firewall oder als Alternative zu Linux das OpenBSD Projekt.
- aber auch dort gibt es Nachteile, wie zB fehlende SMP Faehigkeit und keine Multimedia Fähigkeiten für die Firewall (eigentlich eher ein Vorteil :)

3.) Esoteric/Far−Out Linux Security

Weiterbildung

- eZines lesen: Phrack ist zum Beispiel immer sehr erleuchtend
- der Volksmund sagt das für Security Professionals gilt: Lerne erst Konstruktiv zu Programmieren und wenn du damit genug Erfahrung hast kannst du destruktiv Hacken/Cracken oder andere Bereiche der Sicherheitsthematik ausleuchten.
- sehr zu Empfehlen als aktueller Buchtipp von Bruce Schneier, "Secrets & Lies - Digital Security in a Networked World". Enthält laut neuster Geheimdienstreporte keine mathematischen Formeln und ist damit für jeden Verständlich

Backdoors

- Einer der wichtigsten Gründe für Open Source Software ist die Überprüfbarkeit auf Hintertüren (Backdoors)
- ist faktisch falsch [Doch auch im OpenSource Zeitalter (TM) können in dieser Hinsicht noch böse Überraschungen passieren: In der Borland InterBase Datenbank ist jetzt immerhin schon seit 1994 der geheime Backdoor Account „politically/correct“ vorhanden gewesen. Entdeckt wurde dies erst Anfang 2001.]
- Dies wäre eigentlich sehr leicht zu bemerken gewesen bei einem so schlecht zum Rest des Codes passenden Useraccount. Richtige Backdoors haben aber wohl eher schwer lesebaren Spaghetticode oder super unauffaelligen Code/Namen und dynamisch generierte Passwoerter und keine festen Zeichenketten.

Backdoors

- Good Old Story: Meister der Backdoors ist Ken Thompson gewesen. Sein Hack wäre vermutlich nie bekannt gewesen wenn er ihn nicht bei einem Kongress öffentlich zugegeben hätte
- Er wollte sich die Möglickeit geben in alle Unix Versionen die mit seiner Version des C Compilers erstellt wurden, einzuloggen
- Er schrieb nun ein Stück Code welches bemerkte wenn die Datei login.c kompiliert wurde, und dort falls der Benutzername Ken ist, nichts weiter prüfte. Diesen Code verbarg er direkt im Binary des C Compilers.
- Damit die Backdoor auch bei neuen Versionen des C Compilers erhalten bleibt, musste der C Compiler merken wenn er sich selbst neu kompilierte, und dann den Code fuer den Trigger zur Selbstcompilierung, den Trigger fuer die Login.c Datei und den Check auf den Username Ken wieder einbauen.
- Ich will nicht raten wieviele ähliche Hacks gerade im Umlauf sind.

worst case worm

- Der schlimmste anzunehmende Wurm:
  Testprototyp Samhain von Michael Zalewski. Ist wirklich realisiert worden.
- Eigenschaften:
  - läuft auf verschiedenen Plattformen
  - verbreitet sich unsichtbar
  - verwendet eine verteilte gespeicherte Datenbank von Systemexploits
  - spricht mit anderen Würmern der gleichen Art in einem anonymen, verschlüsselten Freenet ähnlichen „Wurm–Net“
  - verbreitet sich 'tuerlich ohne Input eines Users
  - Payload ist ein Plug–In Modul.
  - desweiteren teilen sich die Würmer neue Exploits, morphen um Antiviren Software zu Verarschen, und würde sich aktiv gegen Debugger und andere ähnliche Werkzeuge wehren

worst case worm

- wie weit das Experiment gekommen ist, ist leider nicht bekannt. Aber eigentlich sollte sich solch ein System mit ein wenig Disziplin und Geld realisieren lassen
- Fragen an das Publikum:
  - Glaubt das Publikum das sowas machbar ist ?
  - Wenn ja, wurde sowas schon von irgendjemandem realisiert ?
  - Egal welche Antwort vorher gegeben wurde, wäre es nicht sowieso egal, da man den Wurm sowieso nicht entdecken könnte ??

Weiterführende Links

- comp.os.linux.security FAQ:
  http://www.memeticcandiru.com/colsfaq.html
- Linux Security HOWTO:
  http://www.linuxdoc.org/HOWTO/Security- HOWTO.html
- Linux Administrators Security Guide:
  http://www.securityportal.com/lasg/
  (sehr praktisch gehalten, mit vielen Themen)
- SecurityFocus unter http://www.securityfocus.com Sehr gutes Security Portal