Zusammenfassung Tag 30

SUID- und SGID-Bits identifizieren

- Das SUID-Bit sorgt dafür, dass ein Programm immer mit Root-Rechten ausgeführt wird, auch wenn es von einem nicht-privilegierten Benutzer aufgerufen wird
- Dies ist z.B. notwendig, um auf geschützte Systemdateien zugreifen zu können, ohne dass den Benutzern selbst der Zugriff darauf gestattet werden muss
- Ein Beispiel hierfür ist das Programm `passwd`, mit dem Passwörter geändert werden können: Auch jedem normalen Benutzer ist es erlaubt, sein Passwort zu ändern. Dazu muss der Passwort-Hash allerdings in `/etc/shadow` geschrieben werden. Diese Datei ist nur von *root* lesbar und veränderbar – daher wird das SUID-Bit für `passwd` gesetzt
- Das SUID-Bit ersetzt das `x` in der dritten Spalte der Rechte des Eigentümers
- Das SGID-Bit wird für die Gruppe gesetzt und sorgt dafür, dass ausführbare Programme neben den Rechten des aufrufenden Benutzers zusätzlich die Rechte der Gruppe erhalten
- Generell sollten insbesondere die Home-Verzeichnisse und Nicht-Standardverzeichnisse nach SUID/SGID-Bits überprüft werden. Wenn sich dort irgendwelche Skripts bzw. Programme verstecken, ist das immer ein Warnzeichen

Benutzerpasswörter und –Konten beschränken

- Eine wichtige Sicherheitsmaßnahme ist traditionell das regelmäßige Wechseln des Passwortes (wird mittlerweile kontrovers diskutiert)
- Diese Änderung kann mit Hilfe der Werte in `/etc/shadow` erzwungen werden
- Das Betriebssystem prüft das Passwort mit Hilfe der Cracklib-Bibliothek auf eine gewisse Qualität
- Um den Zeitraum der Gültigkeit eines Passwortes zu beschränken, kann der Befehl `chage` als `root` genutzt werden
- Das minimale Passwortalter bestimmt, wieviel Zeit in Tagen vergehen muss, bevor ein Benutzer sein Passwort nach einer Änderung erneut ändern kann
- Das maximale Passwortalter legt fest, nach wie vielen Tagen das Passwort geändert werden muss. 99999 steht für "kein Beschränkung"
- Die Passwort-Ablaufwarnung gibt bei der Anmeldung eines Benutzers den Hinweis, dass das betreffende Passwort in X Tagen geändert werden muss
- Passwort inaktiv bedeutet, dass die Anmeldung über diesen Benutzer ggf. auch nach dem Ablauf des Kennwortes möglich ist, er aber innerhalb eines bestimmten Zeitraums spätestens sein Kennwort ändern muss. Danach ist sein Konto gesperrt und nur der Administrator kann es wieder entsperren
- Handelt es sich um einen temporären Benutzer, kann der Zugang zum angegebenen Zeitpunkt gesperrt werden durch ein festes Ablaufdatum für die Anmeldung

Sudo konfigurieren und nutzen

- Unter Ubuntu ist der Benutzer `root` per Default nicht aktiv und Administrationstätigkeiten können nur über `sudo` durchgeführt werden
- Dieses Programm ermächtigt einen unprivilegierten Benutzer, einen Befehl im Kontext des Systemadministrators `root` auszuführen
- Der Vorteil ist, dass dadurch nur normale, unprivilegierte Benutzer für die Anmeldung am System genutzt werden können und somit die Sicherheit des Systems erhöht wird
- Zudem kann sehr granular gesteuert werden, wer welche Rechte hat bzw. temporär als Systemadministrator agieren kann
- Sudo wird durch die Datei `/etc/sudoers` konfiguriert
- Durch `%` wird darin eine Gruppe genannt, die entsprechende Rechte erlangt, z.B. Alle Benutzer, die in der Gruppe sudo sind, dürfen mit sudo alles machen ohne Einschränkungen
- Der Befehl `gentent group sudo` zeigt, wer in dieser Gruppe Mitglied ist
- Standardmäßig wird nur der erste Benutzer, der bei der Installation des Systems angelegt wird, der sudo-Gruppe hinzugefügt. Jeder weitere Benutzer, der als Administrator arbeiten können soll, muss hier manuell hinzugefügt werden
- Die Datei `/etc/sudoers` sollte nie direkt mit einem Editor geändert werden, da eine Fehlkonfiguration dazu führen kann, dass keine Anmeldung mehr möglich ist und das System nur noch im Rescue-Modus betrieben werden kann
- Zur Änderung der Datei wird das Programm `visudo` genutzt. Es enthält eine entsprechende Syntaxprüfung
- Beispiel eines Eintrags:
  `asterix ALL=(ALL:ALL) /usr/bin/ls`
  Damit darf `asterix` den Befehl `ls` im Kontext von `root` ausführen – aber nichts sonst

User-Limits setzen und Aktivitäten überwachen

- Zur Sicherheit kann es sinnvoll sein, Ressourcenlimits für Benutzer zu setzen und zu überwachen, wer derzeit auf dem System angemeldet ist und was er gerade tut
- Der Befehl `ulimit -a` zeigt für verschiedene Bereiche die Limits für den aktiven Benutzer an
- Der Befehl `ulimit -f 0` sorgt dafür, dass der Benutzer zunächst keine weiteren Dateien mehr speichern kann
- Die Einstellung ist immer nur für die aktive Shell gültig. Wird eine neue Shell eröffnet, so sind diese Limits wieder wie zuvor eingestellt
- In `/etc/security` befinden sich diverse Dateien zur Steuerung von Sicherheitsaspekten des Systems
- Die Limits werden in der Datei `limits.conf` konfiguriert
- Alle Zeilen dieser Datei sind per Default auskommentiert

- Es gibt vier Werte: *Domain*, *Type*, *Item* und *Value*
    - Die *Domain* kann ein Benutzer oder eine Gruppe sein. * steht für alle Gruppen werden mit @ gekennzeichnet
    - Der *Type* ist entweder Hard oder Soft
      Soft-Grenzen führen zu Warnungen und Hard-Grenzen zum Sperren der Ressource
    - Welche *Items* (Gegenstände) es gibt, steht oben in den Erläuterungen der Datei
    - der jeweilige *Value* (Wert) bestimmt das Limit
- Der Befehl `who` zeigt alle derzeit angemeldeten Benutzer
- Der Befehl `w` zeigt an, wer am System angemeldet ist und was der Benutzer gerade tut
- Die Historie der Anmeldungen am System zeigt der Befehl `last`

Netzwerk-Dienste identifizieren – netstat und ps

- Durch den traditionellen Befehl `netstat` aus dem optionalen Paket `net-tools` können die aktiven Netzwerk-Dienste des lokalen Systems identifiziert werden
- netstat muss für bestimmte Ausgaben als root ausgeführt werden
- Der Befehl unterstützt diverse Optionen, zum Beispiel `netstat -tlpn`
    - `-t` zeigt die TCP-Ports an
    - `-l` die im Listening-State sind, also auf Anfragen hören
    - `-p` zeigt die dazugehörigen Prozesse an
    - `-n` verzichtet auf die Namensauflösung auf IP- und Portebene
- Der Befehl `ss -tpln` zeigt in ähnlicher Weise die gebundenen TCP-Ports an. Auch hier wird `sudo` benötigt, um die Prozesse anzeigen zu lassen
- Viele Optionen von `ss` sind mit `netstat` identisch
- Formal ist `ss` der neuere Befehl, der genutzt werden soll. Er zeigt geringfügig mehr Ergebnisse an
- Dienste, die von außerhalb kontaktiert werden können, bieten eine potentielle Angriffsfläche
- Gegenmaßnahmen:
    - Dienste entweder deaktivieren, weil sie nicht benötigt werden. Damit ist die Angriffsfläche verschwunden
    - Dienste härten, das bedeutet, sie so zu konfigurieren, dass sie möglichst wenig Angriffsfläche bieten. In vielen Fällen ist die Grundkonfiguration nicht sicher und kann und sollte sicherheitstechnisch optimiert werden
    - Grundsatz: So wenig Funktionalität wie möglich, aber so viel wie nötig bereitstellen
- Der Befehl `ps -ef` zeigt die die Prozessliste
- Unter Umständen können bei genauerer Betrachtung auch hier (Netzwerk-)Prozesse gefunden werden, die eigentlich nicht benötigt werden

- Grundsätzlich sollte man ein Linux-System so schlank wie möglich halten, um die potentielle Angriffsfläche zu minimieren

Offene Ports identifizieren mit Nmap

- Mit dem Portscanner Nmap kann festgestellt werden, welche Ports und Dienste des Zielsystems auf Anfragen reagieren, welche Betriebssystem-Plattform und welche Dienstversionen das Zielsystem nutzt
- Nmap ist ein OpenSource-Tool, das mit vielen gängigen Linux-Distributionen ausgeliefert wird
- Nmap kann unter CentOS installiert werden mit `yum install nmap`
- Nmap startet z.B. mit folgendem Befehl: `nmap 192.168.1.15` (IP des Zielsystems)
  Das Ergebnis zeigt offene Ports des Zielsystems an
- Mit der Option `-sV 192.168.1.1` wird die Versionserkennung der Dienste aktiviert
- Mit `ncat` können Netzwerk-Verbindungen initialisiert oder als Serverdienst sogar auch angenommen werden
- Als Client wird die Adresse des Ziels und, durch Leerzeichen getrennt, die Portnummer angegeben. Beispiel für SMTP-Dienst: `ncat 192.168.1.15 25`
- Reagiert das Ziel im obigen Beispiel, indem ein Banner des Mailservers ausgegeben wird, wurde dieser als solcher identifiziert
- Auf diese Weise kann auch eine Anfrage auf Port 80 erfolgen und mit `GET /` eine Meldung des Webservers provoziert werden. Die detaillierte Angabe enthält oftmals, um welche Software in welcher Version es sich handelt
- Hinter der Versionserkennung von Nmap steckt in einigen Fällen ein einfaches Ablesen der Rückmeldung des Serverdienstes, *Banner-Grabbing* genannt
- Mit der Option `-O` kann Nmap auch noch eine dedizierte Betriebssystemanalyse durchführen
- Je mehr ein Angreifer über ein System in Erfahrung bringen kann, desto eher wird er eine Sicherheitslücke finden, die er gezielt über einen sogenannten Exploit ausnutzen kann
- Da durch Nmap die Perspektive eines externen Angreifers angenommen wird, ist dies ein sehr guter Ansatzpunkt zum Härten der Systeme. Zum Beispiel durch Anpassen des Banners, um eine generische Rückmeldung zu liefern, die nicht sofort Rückschlüsse auf die konkrete Software-Version zulässt

Unnötige Dienste deaktivieren

- Dienste, die nicht verwendet werden, sollten aus Sicherheitsgründen deaktiviert werden
- Status eines Dienstes anzeigen am Beispiel des Druckerdienstes Cups:
  `systemctl status cups`
- Stoppen des Dienstes mit:
  `systemctl stop cups`
- Bei einem Neustart wird der Dienst erneut gestartet

- Damit der Dienst generell nicht mehr gestartet wird, muss er deaktiviert werden: `systemctl disable cups`
- Ein inaktiver Dienst kann trotzdem "enabled" sein und umgekehrt: Ist ein Dienst "disabled", kann er trotzdem derzeit noch laufen, wird aber beim nächsten Neustart nicht mit gestartet
- Aktiviert werden kann ein Dienst über `systemctl enable cups`
- Gestartet wird der Dienst über `systemctl start cups`
- Auf dem System sollte geprüft werden, welche Dienste gestartet und "enabled" sind. Dienste die nicht benötigt werden, sollten entsprechend abgeschaltet werden, um möglichst wenig Angriffsfläche zu bieten

Der Superdaemon xinetd und der TCP-Wrapper

- Der Superdaemon (`inetd`) stellt keinen speziellen Dienst bereit, sondern organisiert diverse, kleine Dienste
- In der Konfigurationsdatei `/etc/inetd.conf` kann der Administrator festlegen, dass bestimmte Netzwerkdienste nicht selbständig über einen eigenen Daemon laufen, sondern vom Superdaemon verwaltet werden
- Der Nachfolger vom `inetd` ist `xinetd`
- `xinetd` hat zwar eine Konfigurationsdatei `/etc/xinetd.conf`, diese enthält aber nur eine Include-Direktive für alle Dateien im Verzeichnis `/etc/xinetd.d`
- Unter `/etc/xinetd.d` steht für jeden von `xinetd` zu verwaltenden Dienst eine eigene Konfigurationsdatei bereit
- Es gibt einige sehr einfache Netzwerk-Protokolle und -Dienste, die standardmäßig über `xinetd` verwaltet werden (z.B. `daytime` oder `echo`)
- `xinetd` bietet einen eigenen TCP-Wrapper (eine Sicherheitskomponente, die den Zugriff auf die jeweiligen TCP-Dienste steuert)
- Über die Dateien `/etc/hosts.allow` und `/etc/hosts.deny` kann bestimmt werden, welche Hosts auf welche Dienste Zugriff haben
    - `hosts.allow` implementiert eine Whitelist, also eine Liste mit erlaubten Zugriffen
    - `hosts.deny` legt dagegen fest, welche Systeme keinen Zugriff haben
    - Trifft ein System auf einen Eintrag in `hosts.deny` zu, ist aber auch in `hosts.allow` eingetragen, so gilt die `hosts.allow`-Direktive
- `xinetd` ist per Default nicht mehr installiert, da heutzutage die Dienste alle über `systemd` organisiert werden
- Nachträgliche Installation ist möglich mit `apt install xinetd` bzw. `yum install xinetd`
- `xinetd` wird bei Ubuntu nach der Installation automatisch gestartet, bei CentOS nicht
- Auf CentOS werden UDP-Dienste als Dgram also Datagram bezeichnet und TCP-Dienste als stream

- Während `xinetd` als Prozess grundsätzlich im Hintergrund läuft, sind die vom Superdaemon verwalteten Dienste per Default nicht aktiv

SSH und die asymmetrische Verschlüsselung

- Die symmetrische Verschlüsselung erstellt einen Schlüssel für die Ver- und Entschlüsselung
- Im Gegensatz zur symmetrischen Verschlüsselung wird bei der asymmetrischen Verschlüsselung ein Schlüsselpaar generiert
- Es besteht aus dem öffentlichen Schlüssel, dem Public Key und dem privaten Schlüssel, Private Key genannt
- Während der private Schlüssel unter maximaler Sicherheit geheim gehalten werden muss, kann der öffentliche Schlüssel ohne Einschränkungen veröffentlicht werden
- Der Public Key wird z.B. dazu verwendet, um einen Klartext zu verschlüsseln, der dann transportiert wird
- Da er verschlüsselt ist, ist es auch grundsätzlich egal, wer den verschlüsselten Text mitlesen kann
- Ein Text, der mit dem Public Key verschlüsselt wurde, kann nur durch den dazu passenden Private Key entschlüsselt werden
- Durch dieses Konzept ist es sehr einfach für einen Kommunikationspartner eine verschlüsselte Nachricht zu versenden, da er lediglich den Public Key des anderen übermittelt bekommen muss
- Das geht auch umgekehrt: Was mit dem Private Key verschlüsselt wurde, kann nur durch den Public Key wieder entschlüsselt werden
- Hier geht es dann nicht um Geheimhaltung, sondern um die digitale Signatur (im Sinne einer fälschungssicheren Unterschrift), also die Sicherstellung, dass die Nachricht tatsächlich von demjenigen kommt, der sich als Absender ausgibt
- Die asymmetrische Verschlüsselung wird primär für zwei Dinge genutzt:
    - Die Übermittlung von Geheimnissen, die niemand lesen können soll
    - Die digitale Signatur, um den Absender zweifelsfrei feststellen zu können
- Die asymmetrische Verschlüsselung ist sehr aufwändig und ressourcenintensiv
- Die symmetrische Verschlüsselungsalgorithmen sind wesentlich effizienter in der Ver- und Entschlüsselung von Daten
- Daher wird in der Regel nur der Verschlüsselungsschlüssel für die symmetrische Verschlüsselung zwischen den Kommunikationspartnern über die asymmetrische Verschlüsselung ausgetauscht und die eigentliche Datenübertragung geschieht mittels symmetrischer Verschlüsselung
- Der SSH-Serverdienst `sshd` kann nachinstalliert werden mit `apt install openssh-server`
- Dienst starten: `systemctl start sshd` und aktivieren: `systemctl enable sshd`
- SSH-Verbindungsaufbau von Client zu Server mit dem Befehl `ssh <Server-Adresse>`

- Beim ersten Verbindungsaufbau wird dem Client der Public Key des Servers übermittelt
- Dieser Moment ist kritisch, da hier entschieden werden muss, ob diesem Schlüssel vertraut werden kann
- Nachdem der Verbindung vertraut wurde, wird der Schlüssel der Datei `/home/user/.ssh/known_hosts` hinzugefügt
- Wird bei einer SSH-Verbindung kein Benutzer angegeben, wird der lokale Benutzer übermittelt
- Mit der Option `-l` kann bei einer SSH-Verbindungsanfrage ein Benutzer angeben werden, mit dem die Anmeldung am SSH-Server erfolgen soll
- Die systemweite SSH-Konfiguration findet sich unter `/etc/ssh`
    - `ssh_config` (Client-Konfiguration)
    - `sshd_config` (Server-Konfiguration)
- Die genannten Dateien enthalten die jeweiligen Public Keys des Servers. Hier finden wir vier der wichtigsten asymmetrischen Algorithmen:
    - RSA – von Rivest Shamir Adleman, sehr beliebt und gut geeignet für Verschlüsselung
    - DSA – Digital Signatur Algorithm, kommt langsam in die Jahre und eignet sich mehr für Signaturen
    - ECDSA – Eliptic Curves DSA, eine Erweiterung von DSA mit besseren Algorithmen
    - ED25519 – eine moderne Form der asymmetrischen Verschlüsselung, sollte bevorzugt genutzt werden
- Der Client fordert den jeweiligen Satz an Algorithmen an und der Server versucht diese Anfrage zu bedienen
- Nur, wenn beide einen übereinstimmenden Satz an Algorithmen finden, kommt die Verbindung zustande
- Im Verzeichnis `/etc/ssh` liegen für jeden der Algorithmen die privaten Schlüssel (ohne Endung) und die öffentlichen Schlüssel (PUB-Datei)

SSH-Key-Authentisierung

- Die Verwendung von Passwörtern ist ein Schwachpunkt in jeder Sicherheitsstrategie
- Eine bessere Form der Authentisierung ist die Verwendung von Public- und Private Keys
- SSH unterstützt diese Form der Authentisierung
- Ein Schlüsselpaar kann auf dem Client generiert werden mit dem Befehl `ssh-keygen -t rsa`
- Dieser hat eine Default-Länge von 2048 Bit
- Das Schlüsselpaar wird in `.ssh` im eigenen Homeverzeichnis abgelegt
- Ein Passwort für den privaten Schlüssel erhöht die Sicherheit, ist aber optional
- Public Key mit `ssh-copy-id benutzer@<Server-Adresse>` auf den Server kopieren

- Im Homeverzeichnis unter `.ssh` befindet sich jetzt eine Datei `authorized_keys`, die Public Keys enthält, die in dieser Form übermittelt wurden und eine automatische Authentifizierung ohne Passwort ermöglichen
- Der Private Key des sich anmeldenden Benutzers wird im Sinne einer digitalen Signatur verwendet, um zu beweisen, dass hinter der Anmeldung tatsächlich derjenige steht, der er vorgibt zu sein
- Unter Windows mit PuTTY (https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html):
    - Mit PuTTYgen, PuTTY Key Generator kann ein Schlüsselpaar erstellt werden
    - Nachdem eine SSH-Verbindung von Client zu Server via PuTTY hergestellt wurde, muss der Public Key in die Datei `.ssh/authorized_keys` kopiert werden
    - In den PuTTY-Einstellungen muss anschließend unter SSH -> Auth der Private Key ausgewählt werden, und im Anschluss kann der Benutzer sich ohne Passwort am Server authentifizieren

Daten verschlüsseln mit GPG – GNU Privacy Guard

- GPG ist ein freies Kryptosystem das hauptsächlich genutzt wird, um E-Mail abzusichern
- Kann auch grundsätzlich sehr einfach für die Ver- und Entschlüsselung von vertraulichen Daten genutzt werden
- GPG ist per Default auf Ubuntu und CentOS installiert
- Mit dem Befehl `gpg –genkey` gelangt man in einen interaktiven Assistenten, um ein Schlüsselpaar zu erstellen
- Das Schlüsselpaar wird im Homeverzeichnis unter `/.gnupg` gespeichert
- Der Befehl `gpg --list-key` zeigt den erstellten Schlüssel und die ID
- Der Public Key muss exportieren werden mit `gpg --export "User-Name" > <user>key.pub`
- Dieser öffentlichen Schlüssel kann nun dem Kommunikationspartner bereitgestellt werden und dort mit dem Befehl `gpg --import benutzerkey.pub` importiert werden
- Der Import kann verifiziert werden durch den Befehl `gpg –list-key`
- Eine Textdatei kann verschlüsselt werden mit dem Befehl `gpg --out geheimtext.enc --recipient benutzer --encrypt klartext.txt`
- Die Datei kann nur der Ersteller des Schlüsselpaares mit dem Private Key entschlüsseln: `gpg --out entschluesselt.txt --decrypt geheimtext.enc`
- Das Prinzip der asymmetrischen Verschlüsselung wird hier im Prinzip genauso eingesetzt wie bei SSH