Linux

Advanced Security

Labor-Umgebung

Für diese Schulung sind entwender **lokale VMs** oder ein **Cloud-Labor** vorgesehen.

Hostname	**IP (*Vagrant*)**	**IP (*Cloud*)**	**Rolle**
`controller`	192.168.56.10	TBA	Controller
`node1`	192.168.56.20	TBA	Node 1
`node2`	192.168.56.30	TBA	Node 2

Der Host `controller` dient als Jumphost - es gibt verschiedene unterschiede **Zugangsdaten**:

Benutzername	**Passwort**	**Beschreibung**
`user`	`SVA2024-SCgLKzeyj9v4maXsxqJuWD`	SSH, alle Hosts
`root`	`SVA2024-sHv9jUtAJR5hTgfdZwKa8S`	SSH, alle Hosts
`admin`	`SVA2024-F9RnfQr2pZjYEVqWJUgNeH`	OpenVAS-User

In der Vagrant-Umgebung erfolgt der Login wie folgt:

```
$ vagrant ssh controller
$ sudo su - user
```

Für den Login in der Cloud-Umgebung sind die folgende Kommandos notwendig:

```
$ ssh user@<ip-adresse>
```

Es empfiehlt sich anschließend den SSH-Schlüssel auch auf die beiden Server `node1` und `node2` zu kopieren - so erspart man sich das Eintippen des Passworts im weiteren Verlauf der Schulung:

```
$ ssh-copy-id node1
$ ssh-copy-id node2
```

Vagrant

Das Labor kann später auch zu Lernzwecken auf dem **eigenen Rechner** aufgesetzt werden. Hierfür werden benötigt:

- HashiCorp Vagrant (https://www.vagrantup.com)
- Oracle VirtualBox (https://www.virtualbox.org)
- 8 GB Arbeitsspeicher
- mindestens 40 GB Festplattenspeicher

Der Schulung liegt ein Ordner bei, der die entsprechenden Konfigurationen enthält.
Mithilfe von Vagrant können die drei VMs vollautomatisiert erstellt und konfiguriert werden.

Zur Bereitstellung eine Kommandozeile öffnen und in den **Schulungsordner** wechseln:

```
$ cd Vagrant
$ vagrant up
```

Die Bereitstellung kann bis zu **30 Minuten dauern**, es wird ein Rocky Linux 8-Template heruntergeladen.

Musterlösungen

Für die einzelnen Aufgaben der Schulung gibt es im Unterordner `labs` Musterlösungen. Auf den Labor-Systemen sind diese Dateien unterhalb des Ordners `/labs` erreichbar.

Die Lösungen bestehen entweder aus einem Skript oder einem Ordner, in welchem sich ein Skript befindet. Um die Lösung anzuwenden, genügt es in den Ordner zu wechseln und das Skript auszuführen.

```
$ cd /labs/01
$ bash node1.sh
```

Kapitel-Zusammenfassungen

Grundlagen

- eine **minimale** Paketauswahl aus **vertrauenswürdiger** Quelle ist sinnvoll
- automatische Updates
- Sicherheitsmechanismen (*Firewall, SELinux*) **nicht deaktivieren**
- Dateisystem-Berechtigungen regelmäßig **überprüfen**
- sinnhaftige Passwort-Richtlinien festlegen
- SSH-Zugriff **erschweren**
    - kein Root-Zugriff
    - Schlüssel- statt Kennwort-Authentifizierung
- Rootkit-Analyse durch Einsatz von **AIDE** o.ä. erschweren

SELinux

- **Linux Kernel Security Modules** ermöglichen den Einsatz weiterer Sicherheitsmodule
    - u.a. AppArmor und SELinux
- SELinux ergänzt Linux um **M**andatory **A**ccess **C**ontrol
    - definiert welche Ressourcen von Programmen und Dienst genutzt werden dürfen
    - unterbindet Verstöße auf **höchster Ebene**
    - alles, was nicht **explizit erlaubt** ist, ist verboten
- **M**ulti-**L**evel **S**ecurity kann optional um **Schutzklassen** ergänzen
    - z.B. Streng geheim, Geheim, Vertraulich, Öffentlich
- SELinux ist vor allen auf **Red Hat**-artigen Distributionen beliebt

    - erkennt und steuert Zugriffe auf Prozesse, Netzwerk-Ports und das Dateisystem
    - wird vor allem durch Ressourcen-Kennzeichnung (**Kontext**) erreicht
- Ein **Kontext** besteht aus einem **User**, einer **Rolle** und einem **Typ**
    - es existiert eine dedizierte User-Datenbank
    - Rollen sprechen Usern Berechtigungen zu
    - Typen verbinden Resourcen mit Prozessen (**Domains**)
- SELinux-**Policies** beinhalten Zugriffs-/Verbotsregeln
    - **Booleans** können das Verhalten beeinflussen
- Verschiedene Tools können beim **Troubleshooting** helfen
    - `ausearch`
    - `setenforce`
    - `audit2why`
- SELinux und AppArmor sind nur **Teilkomponenten** eines sicheren Systems
    - sie ersetzen weder Firewalls noch Antiviren-Software

AppArmor

- steuert ebenfalls Prozesse auf **Systemebene**
    - Ausnahmen werden explizit **pro Prozess** aktiviert
- implementiert **Access Control** u.a. für
    - Dateien, Capabilities, Netzwerk, Mounts, DBUS und Unix Sockets
- **Profile** definieren Berechtigungen einer Anwendung
    - Anwendung darf nur erlaubte Aufrufe tätigen
    - können modular gehalten werden, um die Pflege zu erleichtern
    - **Flags** geben Berechtigungen an
- Variablen werden als **Tunables** gespeichert
- Mit **Abstractions** existieren wiederverwendbare Vorlagen

OpenVAS

- **Scanner** wie OpenVAS können beim Aufdecken von **Verwundbarkeiten** helfen
    - oft im Zusammenhang mit **Penetrationstests** verwendet
- Software-**Framework** zur Ausführung umfangreicher Überprüfungen
    - überprüft u.a. Server-Dienste auf **Sicherheitslücken**
    - kann **Netztraffic** analysieren, um verwendete Komponenten zu erraten
    - startet i.d.R. ohne Interaktion mit den betroffenen Systemen
- Zu überprüfende Hosts werden als **Ziele**
    - gefundene Lücken können dokumentiert und verwaltet werden

    - **Scans** können **wiederkehrend** stattfinden

fail2ban

- Intrusion Detection/Prevention Systeme können vor **Bruteforcing** schützen
- **fail2ban** integriert sich in verschiedene **Komponenten**
    - Firewalls, Mail-Server, Webserver, Datenbanken und Applikationen
- **Filter** kontrollieren **Log-Dateien** überwachter Dienste
    - Suche anhang **Schlagwörtern** und regulärer Ausdrücke
- **Actions** dienen zur Sperrung nach erreichten **Schwellwerten**
- Das Zusammenspiel aus Filter und Actions wird auch **Jail** genannt
- Reporting via Mail oder Integration in gängige Monitoring-Lösungen

Dev-Sec

- Systeme müssen **regelmäßig auditiert** werden
    - ständige Updates bringen **neue** Komponenten und somit auch **Lücken**
- Hardening muss weitestgehend automatisiert werden > **praktikabel**
- Dev-Sec **automatisiert** das Auditieren und Absichern von Systemen
- vorgefertigte **Baselines** und Härtungs-Automatismen u.a. für
    - Linux, SSH, Apache
    - Inhalte auf Basis von CIS-Benchmarks und NSA-Hardening Guides
- Ausgiebiges **Testen** unabdingbar
- `ansible-lockdown` ist eine weitere Alternative