BAI4-RN	Praktikum Rechnernetze – Aufgabe 2	MDE/KSS
SoSe 2024	Entwicklung eines einfachen Client/Server-Systems – nunmehr im Netzwerk	

Vorbereitung

Für ein erfolgreiches Praktikum sollten Sie sich anhand der im Internet verfügbaren Informationen über folgende Anwendungen informieren:

- SSH-Clients für Ihr Endgerät
  - für Windows: PUTTY bzw. WINSCP
- „Durchtunneln“ von Zugriffen durch einen SSH-Server, so dass auf den Server „dahinter“ zugegriffen werden kann (also die TCP-Verbindung zwar über den SSH-Server geführt wird, aber die lokale Anwendung direkt mit dem remote SSH-Server kommuniziert und „dort“ gearbeitet wird)
- Nutzung von SSH Public Keys, um ohne Passworteingabe Zugriff zu erlangen
- Sicherlich müssen Sie auch Zugriffsrechte unter LINUX/UBUNTU anpassen und Netzwerkdienste im Betriebssystem verankern, wobei erfahrungsgemäß bei vielen praktische Erfahrung fehlt. Also bitte ggf. auch dazu etwas lesen.
- Patchen lokaler Anwendungen auf einem Linux-Server

Im Praktikum werden Ihnen Maschinen und Identifier für Ihre Arbeitsgruppe zugeteilt. Wenn Sie nach Ihrem ersten Termin keine haben, bitte melden!

Warnhinweis!

Da die Aufgaben auf einem Virtualisierungsserver innerhalb der HAW gelöst werden sollen, zumindest der Teil, den Sie nicht lokal machen, steht Ihnen die virtuelle Maschine zur Verfügung. Ihre virtuellen Maschinen befinden sich auf der gleichen Hardware, sind aber weitestgehend von denen anderer unabhängig.

Allerdings gibt es auch keine zentrale IT-Abteilung, die Ihnen die lästigen Aufgaben wie Backup und Maintenance abnimmt. Tatsächlich kann es vorkommen, dass die Maschine von heute auf morgen nicht mehr verfügbar ist. Nicht, weil wir das als Schikane in der Mitte des Semesters mal machen, um zu prüfen, ob Sie dies hier gelesen haben, sondern weil eben ein Hardware-Ausfall jederzeit passieren kann, oder ein Mensch einen Fehler macht und das System ihre Maschine „vergessen“ hat. Hierbei könnte es eben auch zu Datenverlusten kommen.

Und um das Ganze noch konkreter auszudrücken: Mitunter sperren sich Studierende (und auch Profs oder wissenschaftliche Mitarbeiter ;) selbst aus. Dann kann man evtl. nichts mehr machen (z.B. verhindert die lokale Firewall ALLE neuen Verbindungen zu Server-Prozessen auf der Maschine, sobald man sich ausloggt, ist es vorbei. Und an die Konsole zu kommen, ist uns eben NICHT MÖGLICH!). D.h. Sie verlieren alles, was Sie sich nicht lokal gesichert haben.

Sie sind verantwortlich dafür, dass Sie selbst Ihre Ergebnisse zwischenspeichern und so archivieren, dass Sie von solchen „Katastrophen“ unbeeinträchtigt weiterarbeiten können!

Aufgabe 2a: Zugang zur Linux-Maschine

Auf dem Virtualisierungsserver bekommt jede Arbeitsgruppe eine eigene Linux-Maschine incl. IP-Adresse zur Verfügung gestellt. Für die IP-Adresse gibt es keine Namensauflösung über DNS. D.h. lokal auf Ihren Geräten können Sie die Namensauflösung durch Einfügen eines Eintrags (bei Linux: `/etc/hosts`) erreichen. Die Nummer Ihrer zugeteilten Arbeitsgruppe „01“ bis ... ist wichtig und muss an der richtigen Stelle – gekennzeichnet durch das „XX“ im unten angegebenen Hostname – eingesetzt werden:

- `Hostname:`
  `svsXX.ful.informatik.haw-hamburg.de`
- `Accountname:`
  `padawan`
- `Passwort:`
  `PassierscheinA38`

Sie können aus dem Netz der HAW (über WLAN bzw. VPN) später Web-Anwendungen auf Port 80/tcp (HTTP) und 443/tcp (HTTPS), die auf Ihrer Maschine laufen, erreichen. Dies gelingt NICHT aus dem Internet!
Weil der Zugang zu der virtuellen Maschine über Port 22/tcp (SSH) noch besonders geschützt ist, müssen Sie sich dafür immer zunächst mit Ihrer inf-Kennung auf dem SSH-Gateway (hop-ful.informatik.haw-hamburg.de, IP 141.22.40.11) anmelden. Um zu prüfen, ob Sie auf dem richtigen SSH-Gateway gelandet sind, helfen diese Daten:

```
ECDSA key fingerprint is
AAAAC3NzaC1lZDI1NTE5AAAAIBw8LZLIQ4RDOfez1+BMQ00+Dr6Krrx8YzTE/WWxvptf
```

(Dies ist ein sogenannter kryptographischer Hash, der an die eingesetzten Schlüssel gebunden ist. Wenn Sie wieder zu dem SSH-Server sich verbinden, und Ihr Client bemerkt, dass der aktuell verbundene SSH-Server einen anderen Fingerprint „vorzeigt“, erhalten Sie eine Sicherheitswarnung! Keinesfalls sollten Sie dann – ohne geklärt zu haben, dass es der richtige Server ist – dort ein Passwort eingeben!)

Der Zugriff auf die Maschinen erfolgt über SSH, durch die Verwendung individueller Public-Keys soll ein authentisierter und vertraulicher Zugriff möglich sein. Hierzu muss mit einem lokalen Client ein solcher individueller Public-Key zunächst erzeugt und auf der Maschine in „Ihrem“ Account in der Datei „`authorized_keys`“ abgelegt werden. Sie können auch mehrere öffentliche Schlüssel in diese Datei kopieren, so dass alle in der Gruppe leicht Zugriff erlangen. Beachten Sie, dass die Syntax der Schlüssel zwischen UBUNTU/LINUX und PUTTY definitiv anders ist, Sie also die Daten vor dem Hineinkopieren anpassen müssen (gilt evtl. auch für andere Clients, also aufpassen und prüfen!).

Sie sollten – das gilt immer – einen Ihnen zugewiesenes Passwort schnell ändern! Auch wenn Sie den Zugang mit Ihrem persönlichen SSH Private-Key möglich machen, ist es clever, ab und zu selbst einen zweiten Zugang zu haben und definitiv nur selbst über das dafür verwendete Passwort zu bestimmen. Im Betrieb müssen Sie aufpassen, dass der zweite Zugang genauso sicher ist und nicht die Angreifer den SSH-Zugang ignorieren und Ihr Test-Passwort einfach raten 😉

In dieser Praktikumsaufgabe geht es zunächst darum, dass Sie überhaupt ein System haben, auf dem Sie uneingeschränkt als „`root`“ arbeiten können, ohne Gefahr zu laufen, alles kaputt machen zu können. Dies kann am einfachsten über das zur Verfügung gestellte Grundgerüst im Dockerfile gewährleistet werden:

```
Teams -> Praktikum (Allgemein) -> Dateien ->
snakeoil.tar.gz
```

Für die Vorbereitung der ersten Schritte bedarf es lediglich eines Downloads des Projekt-Archivs und der Konfiguration des Projekt-Namens, welcher die Gruppennummer repräsentiert. Dies erfolgt in der Dockerfile zur environment variable `CI_PROJECT_NAME`.

Der beiliegenden README.md können Hilfestellungen entnommen werden.

Für die Abgabe dieser Teilaufgabe ist keine besondere Aktion vorgesehen. Sie benötigen allerdings eine funktionierende Umgebung, damit Sie weiter erfolgreich arbeiten können.

Aufgabe 2b: Einrichten Ihres einfachen Servers

Richten Sie Ihre Maschine und Ihre Arbeitsplätze so ein, dass ein Zugriff über Port 80/tcp auf „Ihren“ TCP-Server möglich wird. Verbinden Sie sich mit Ihrem Arbeitsplatz in das HAW-Netz und benutzen Sie „Ihren“ TCP-Server.

Warum Port 80/tcp? Weil dieser Port für das Dockerfile und die Firewalls in der HAW freigegeben ist. D.h. Ihr Client holt sich eine lokale Portnummer (im hohen Bereich), darf aber nur bestimmte remote Ports erreichen – auf den virtualisierten Systemen eben 80/tcp und 443/tcp. Da der letzte für HTTPS und sichere Verbindungen steht – was bei Ihren Systemen nicht der Fall ist – nehmen wir einfach Port 80/tcp. Das ist gegen die Konvention, aber erlaubt.

Überprüfen Sie, dass alles läuft, was auch bei Ihnen lokal lief!

Aufgabe 2c: Änderungen am Server

Wenn der Server mit dem Client interagiert, geben die meisten von Ihnen nicht viele Daten aus. Wenn alles läuft, ist das üblicherweise auch kein Problem, aber es läuft ja immer irgendwas nicht richtig und deswegen gilt es immer, vorzubeugen.

Stellen Sie sicher, dass Ihr Server auf der Konsole sinnvolle Ausgaben ausgibt:

- Wenn eine neue Verbindung aufgebaut wird und was mit der Verbindung passiert.
- Wenn ein neues Kommando kommt, über welche Verbindung es kommt, was es war und wie die Reaktion darauf war.
- Wenn ein Shutdown-Kommando kommt, weitere Hinweise, was beim Server „abgeht“.

Achten Sie darauf, dass eine Ausgabe auf der Konsole „gefährlich“ sein kann. Es gibt Zeichen, die auf der Konsole nicht dargestellt werden, aber doch eine Funktion haben könnten. Kodieren Sie nicht druckbare Zeichen wie es z.B. auf einem Linux-System der folgende Befehl in der Ausgabe (\n anstelle eines Zeilenwechsels) machen würde:

```
user% od –c /etc/motd

0000000    L   i   f   e       i   s       b   a   s   i   c   a   l   l
0000020    y       g   o   o   d  \n  \n
0000030
```

Bei richtigen Servern erfolgt auch das Logging über das Netzwerk. Dazu werden die Nachrichten mit Priorität und Quelle versehen an einen SYSLOG-Server übergeben (den wir jetzt nicht im Netz aufbauen). Damit Ihre Programme dies später leicht nutzen können, empfiehlt es sich, von Anfang an durch entsprechende Funktionen die Schnittstellenanpassung leicht zu machen. Für JAVA gibt es leicht nachzuvollziehende Beispiele, z.B.

https://alvinalexander.com/java/jwarehouse/syslog-1.2/com/ice/syslog/Syslog.java.shtml

D.h. nutzen Sie eine Funktion, die alle Parameter verwendet, die auch beim SYSLOG-Aufruf benötigt werden (Facility, Priority, Message/msg), nur dass Sie dann etwas auf der Konsole ausgeben:

```
public void
        syslog( int facility, int priority, String msg ) { … }
```

siehe auch: https://man7.org/linux/man-pages/man3/syslog.3.html

Aufgabe 2d: Änderungen am Client

Bevor Sie mit Ihren Änderungen am Client in die Tests gehen, stellen Sie sicher, dass Ihr „alter“ Client mit dem neuen Server (aus Aufgabe 2c) aus Ihrer Sicht wieder einwandfrei funktioniert. Denken Sie auch daran, dass Sie eine ganze Reihe von Server-Tests durchlaufen lassen können, ohne dass Sie dazu Ihren Client brauchen:

```
user% nc svsXX.ful.informatik.haw-hamburg.de 80 < test.data
```

Bauen Sie in den Client ein, dass die Anzahl der Bytes, die MAXIMAL in einem Netzwerk-Paket gleichzeitig zum Server geschickt wird, beim Aufruf des Clients oder als Option bei der Nutzung vorgegeben werden kann. Was ist hierbei zu beachten?

- Mindestens ein Byte an Nutzdaten muss pro Paket übergeben werden.

Führen Sie dann wieder alle Tests durch, zunächst mit einem Wert von 255, dann mit 10 und zuletzt mit 1, und beobachten Sie für alle Tests, wie der Server reagiert. Wenn Sie die Aufgabe 2c gut gemacht haben, sehen Sie auch, was passiert. Und haben gleichzeitig eine Dokumentation über alle Testläufe 😉

Es ist zu erwarten, dass das, was passiert, nicht Ihren Erwartungen entspricht. Die Chancen sind 90%, dass die Kommunikation nicht fehlerfrei funktioniert, spätestens wenn Sie beim Wert 10 Nachrichten schicken, die länger sind als 10 Zeichen oder eben bei dem Wert 1. Finden Sie die Fehler im Server und passen Sie die Verarbeitung dort an, so dass wieder alles funktioniert!

Was muss alles funktionieren?

- Alle in der Arbeitsgruppe haben einen SSH-Zugang zu der virtualisierten Maschine von Ihren Arbeitsplätzen aus und das initiale Passwort ist geändert.
- Server funktioniert auf der virtuellen Maschinen und läuft.
- Egal, was der Server über das Netzwerk bekommt, gibt es sinnvolle Ausgaben, anhand derer auch Leute, die den Server nicht programmiert haben, nachvollziehen können, was „passiert“.
- Der Server kommt auch dann mit der Kommunikation zurecht, selbst wenn der Client nur jeweils ein Byte an Nutzdaten gleichzeitig schickt.

Eigentlich muss zwischen Client und Server wie vorher (aber besser im Sinne defensiver Programmierung) alles funktionieren, was spezifiziert war. Um das zu testen, was wir „Interoperabilität“ nennen, können wir jetzt quasi alle Clients mit allen Servern testen und schauen, ob auch da – in der Kombination – alles klappt 😉

Dazu mehr im Praktikum!