Documentation

Content

- Network
- Django
- Hardening
- Analyse

Dokumentation Heimnetz

Home

Aufgaben Durchführen Teil 1

- ☒ Netze rot, grün, orange entsprechen eigenen Vorgaben
  - ☒ IP
  - ☒ Verbindung Host
  - ☒ Verfügbarkeit der Dienste
- ☒ IPFire, Server und Adminrechner folgen eigenen Vorgaben
- ☒ IPFire aufgesetzt, Ping möglich
- ☒ DNS von IPFire funktioniert entsprechend Vorgabe
- ☒ DNS Server im Intranet aufgesetzt und konfiguriert
- ☒ DNS löst bei Test auf Adminrechner lokale und globale Namen auf
- ☒ Interner DHCP Server im Intranet aufgesetzt und konfiguriert
- ☒ Web-Server aufgesetzt und konfiguriert
-

[x] Webserver mit Firefox des Hosts aufrufbar mittels IP (http)

Zusatz

- ☒ Zugriff auf Webserver über Hostname statt IP ist möglich
- ☒ Zugriff auf Webserver über Hostname und SSL/TLS verschlüsselt (https)
- ☒ Proxyeinstellung auf IPFire erlauben https Zugriffe von IPFire ins Internet
- ☒ Kommunikation srv04web mit srv03db

IP-Fire

- Hostname: srv01ipfire
- Hinzufügen der drei Switche
- Installation wie gewohnt
  - Hinzufügen der MAC-Adressen der Switche
    - VMnet8 - 00:15:5D:B2:14:07
    - VMnet1 - 00:15:5D:B2:14:09
    - VMnet2 - 00:15:5D:B2:14:08
  - Vergabe IP-Adressen
    - VMnet8 = 192.168.72.3
    - VMnet1 = 192.168.13.3
    - VMnet2 = 192.168.113.3
- DNS -> 192.168.72.2

Proxy-Einstellungen

```
$ nano /etc/profile

export http_proxy=http://10.254.5.100:3128
export https_proxy=http://10.254.5.100:3128
export no_proxy=localhost,srv04web.doubtful-joy13.de,192.168.113.20
```

Bild Proxy

unbound.conf

- srv01ipfire -> `/etc/unbound/unbound.conf`

```
# Insecure Domains
domain-insecure: bsz-et.lan.dd-schulen.de

# Local Zones
unblock-lan-zones: yes
insecure-lan-zones: yes

# Hardening Options
harden-dnssec-stripped: no
harden-large-queries: yes
harden-referral-path: yes
aggressive-nsec: yes
```

Firewall-Regeln

Begründung + Port 80 für unsicherern Webzugriff + Port 443 für Webzugriff über TLS + Port 3306 für Zugriff auf die Datenbank aus der DMZ

DNS + DHCP

- Hostname: srv02dc
- Hinzufügen Switch intern 1 für VMnet1
- Betriebssystem: CentOS 9
- Konfiguration statische IP-Adresse:

```
nmcli device
nmcli connection modify eth0 ipv4.addresses 192.168.13.20/24
nmcli connection modify eth0 ipv4.gateway 192.168.13.3
nmcli connection modify eth0 ipv4.dns "8.8.8.8 8.8.4.4"
nmcli connection modify eth0 ipv4.method manual
```

- Installation `dnsmasq`

```
yum -y install dnsmasq
systemctl start dnsmasq
systemctl enable dnsmasq
systemctl status dnsmasq
```

Bearbeitung der /etc/dnsmasq.conf

```
$ cp /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
$ nano /etc/dnsmasq.conf

listen-address=127.0.0.1,192.168.13.20
interface=eth0
expand-hosts
domain=Doubtful-Joy13.de
server=192.168.72.2
server=8.8.8.8
server=8.8.4.4
address=/Doubtful-Joy13.de/127.0.0.1
address=/Doubtful-Joy13.de/192.168.13.20

$ dnsmasq -test
$ nano /etc/resolv.conf

nameserver 127.0.0.1
```

- die `resolv.conf` wird vom lokalen deamon verwaltet (u. a. NetworkManager), welcher Änderungen überschreibt
- daher wird sie schreibgeschützt

```
chattr +i /etc/resolv.conf
lsattr /etc/resolv.conf
```

- die `/etc/hosts` bearbeiten, um dnsmasq-controller, gateway, Webserver und Datenbankserver einzutragen

```
nano /etc/hosts
        127.0.0.1   srv02dc.doubtful-joy13.de srv02dc
        192.168.13.20   srv02dc.doubtful-joy13.de srv02dc
        192.168.13.3    srv01ipfire.doubtful-joy13.de srv01ipfire
        192.168.113.20  srv04web.doubtful-joy13.de srv04web
        192.168.13.22   srv03db.doubtful-joy13.de  srv03db
```

- Testen der lokalen Domäne `Doubtful-Joy13.de`

```
Bemerkung: ipfire wurde umbenannt von "gateway" in "srv01ipfire"
```

Firewall-Regeln

```
firewall-cmd --add-service=dns --permanent
firewall-cmd --add-service=dhcp --permanent
firewall-cmd --reload
```

Zusatz: Aufruf Webserver über Hostname

```
# Setting up A Record for srv04web

$ nano /etc/dnsmasq.conf
  host-record=srv04web.doubtful-joy13.de,192.168.113.20

# Adding in C:\Windows\System32\Drivers\etc\hosts
```

Einrichtung DHCP

- Bearbeitung der `/etc/dnsmasq.conf`

```
dhcp-range=192.168.13.50,192.168.13.150,12h
dhcp-leasefile=/var/lib/dnsmasq/dnsmasq.leases
dhcp-option=3,192.168.13.3
dhcp-option=option:dns-server,192.168.13.20
dhcp-option=option:netmask,255.255.255.0
```

Installation Administrator-Rechner

- Hostname: cl01admin
- Hinzufügen Switch intern 1 für VMnet1
- Betriebssystem: CentOS 9 -> diesmal als Client-Rechner
- gewohnte Installation
- LAN-Einstellungen > IPv4-Methode > Automatisch DHCP
  - Rechner erhält nun über aufgesetzten DHCP-Server seine IP-Adresse, die Subnetzmaske und das Gateway

Installation Datenbank-Server

- Hostname: srv03db
- Hinzufügen Switch intern 1 für VMnet1
- Betriebssystem: CentOS 9 -> Server
- gewohnte Installation
- da der Server die Daten vom Webserver erhalten soll, benötigt er eine statische IP vom DHCP

```
srv02dc$ nano /etc/dnsmasq.conf
dhcp-host=00:15:5d:b2:14:0e,srv03db,192.168.13.22
```

- Auflösung von DNS-Anfragen funktionieren -> geprüft mit `ping google.de`

Installation MariaDB

```
yum install mariadb-server
systemctl enable mariadb
systemctl start mariadb
mysql_secure_installation
  Enter
  Yes
  No
  Yes
  Yes
  Yes
  Yes
mysql -u root -p
  CREATE DATABASE tickets;
  CREATE user admin;
  GRANT ALL ON tickets.* TO admin@srv04web IDENTIFIED BY '12345';
  exit;
```

Firewall-Richtlinien

```
$ firewall-cmd --zone=public --add-port=3306/tcp
$ firewall-cmd --runtime-to-permanent
```

Konfiguration SELinux für Datenbankzugriff

```
$ setsebool -P httpd_can_network_connect_db on
```

Installation Webserver

- Hostname: srv04web
- Hinzufügen Switch intern2 für VMnet2

Installation http (apache)

```
yum install httpd
systemctl start httpd
systemctl enable httpd
firewall-cmd --permanent --zone=public --add-port=80/tcp
firewall-cmd --permanent --zone=public --add-port=443/tcp
```

Installation php8.1

```
$ dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm

$ dnf install https://rpms.remirepo.net/enterprise/remi-release-9.rpm

$ dnf module install php:remi-8.1

$ php -v
```

Zusatz: SSL/TLS-Verschlüsselung

```
yum install mod_ssl
```

Proxy auf allen Servern (außer IPFire) einrichten

- ☒ srv02dc
- ☒ srv03db
- ☒ cl01admin
- ☒ srv04web

```
$ nano /etc/environment

export http_proxy=http://10.254.5.100:3128
export https_proxy=http://10.254.5.100:3128
export no_proxy=localhost,srv04web.doubtful-joy13.de,192.168.113.20
```

Überprüfung der Verbindungen // Firewall-Regeln

- ping an srv01ipfire von allen Maschinen möglich
- ping zwischen den Maschinen im Grünen Netz möglich
- ping von srv04web ins Grüne Netz nicht möglich -> muss auch nicht möglich sein
- ping aus dem Grünen Netz zu srv04web -> funktioniert

Durchführen Teil 2

- ☒ Ablaufdoku mit aktuellem Netzwerkplan als PDF im Ordner abgelegt
- ☒ Datenbankserver im richtigen Netz aufgesetzt und updatebar

- ☒ Administrativer Zugriff auf DB-Server und Web-Server vom Adminrechner aus möglich
  - ☒ Administrativer Zugriff über SSH-Key (root) möglich

```
# Erstellung key-pair auf cl01admin

$ ssh-keygen
/home/admin/.ssh/id_rsa

# Übertragung des public key auf srv03db srv04web

$ ssh-copy-id root@srv03db

$ ssh-copy-id root@srv04web
```

- ☒ Datenbank und Datenbankadmin angelegt, lokaler Zugriff möglich
- ☒ Datensatz, der auf Web-Server erfasst wird erscheint im DB Server (Nachweis mit SQL Befehl auf dem DB Server)

```
mysql -u root -p
show databases;
use tickets;
select * from ticketwand_ticket
```

- ☒ Kommunikation DMZ -> Intranet ist in IPFire auf das Notwendige beschränkt
  - nur Kommunikation/Port für Datenbank ist geöffnet (3306)

Deploy Django

```
dnf install mod_wsgi httpd
systemctl restart httpd.service

# env

python3 -m venv .env
source .env/bin/activate
pip install -r lf9-feuerwand/env.conf


# /etc/httpd/conf.d/python-wsgi.conf

# settings.py

ALLOWED_HOSTS = ["192.168.72.3"]

DATABASES = {
```

```
  'default': {
    'ENGINE': 'django.db.backends.mysql',
    'NAME': 'tickets',
    'USER': 'admin',
    'PASSWORD': '12345',
    'HOST': '192.168.13.22',
    'PORT': '3306',
  }
}
```

Auswertung und Reflexion

- ☒ Wesentliche Schritte der Installation und Konfiguration der Services für DNS/DHCP, Webserver/Programmiersprache/DB sind in der Ablaufdoku beschrieben.
- ☒ Geänderte und ergänzte Eintragungen in den Config-Files sind angegeben und kommentiert.
- ☒ Firewalleinstellungen von IPFire sind in der Ablaufdoku beschrieben und begründet.
- ☒ Aktueller Netzwerkplan dokumentiert den finalen Aufbau
- ☒ Soll-Ist-Vergleich: Ergebnis und Zeitaufwand
- ☒ Benennung von Defiziten zur Sicherheit der Lösung - Optimierungsvorschläge

Documentation for Django

Home

Content

- Views
  - IndexView
  - LoginView
  - CreateTicketView
- Models
  - Ticket
- Forms
  - CreateTicketForm

Views

IndexView

```
class IndexView(View):
    """
    Index
```

```
    """

    def get(self,request:HttpRequest)->HttpResponse:
        """
        gets
        """
        return render(request,"ticketwand/index.html")
```

Methods

- `get`
  - redners template

Templates

- `index.html`

Tests

- no

back

LoginView

```
class LoginView(LoginView):
    """Login"""

    template_name = "ticketwand/login.html"
    success_url = "{% url 'index-view'%}"
```

Special

- `LoginView` from Django

Templates

- `login.html`

back

CreateTicketView

```
class CreateTicketView(View):
    """Create Ticket"""
```

```
    def get(self,request:HttpRequest)->HttpResponse:
        """get"""
        form = CreateTicketForm()
        return render(request,"ticketwand/create_ticket.html",{"form":form})

    def post(self,request:HttpRequest)->HttpResponse:
        """post"""
        form = CreateTicketForm(data=request.POST)
        if form.is_valid():
            text = form.cleaned_data["text"]
            title = form.cleaned_data["name"]
            email = form.cleaned_data["email"]
            Ticket.objects.create(name=title,text=text,creator_email=email,since=datetime.no
            return render(request,"ticketwand/ticket_created.html")
        return redirect("create-ticket-view")
```

Methods

- `get`
  - redners template with `CreateTicketForm`

- `post`
  - checks if `CreateTicketForm` is valid
  - redirects if form is invalid
  - creates `Ticket` if form is valid

Templates

- `ticket_created.html`
- `create_ticket.html`

Tests

- no

back

Models

Ticket

```
class Ticket(models.Model):
    """Model for Ticket"""

    status_choice = (
    ("in Bearbeitung", "in Bearbeitung"),
    ("nicht lösbar", "nicht lösbar"),
    ("reserveiert", "reserviert"),
```

)

```
    name = models.CharField(max_length=250,null=False,default="")
    text = models.TextField(null=False,default="")
    since = models.DateTimeField(null=False)
    creator_email = models.EmailField(null=False)
    status = models.CharField(null=True,choices=status_choice,max_length=200)
    owner = models.ForeignKey(User, on_delete=models.CASCADE,null=True)
    is_active = models.BooleanField(null=False,default=True)
    labels = models.TextField(null=True)
```

Fields

- `name` -> ticket headline
- `text` -> ticket text
- `since` -> creation date of the ticket
- `creator_email` -> email of the user who created the ticket
- `status` -> status of the ticket, see `status_choice`
- `owner` -> FK to django user model
- `is_active`-> bool if ticket is active
- `labels` -> textfield to be filled with labels

Tests

- no

back

Forms

CreateTicketForm

```
class CreateTicketForm(forms.Form):
    """Create Ticket form"""

    name = forms.CharField(max_length=250,required=True)
    text = forms.Field(required=True)
    email = forms.EmailField(required=True)
```

Fields

- `name` -> ticket headline
- `text` -> ticket text
- `email` -> email of the user who created the ticket

Tests

- no

back

Soll-Ist-Vergleich

Home

Soll	Ist
Doubtful-Joy fordert eine Segmentierung der Netzinfrastruktur mit einer sicheren Trennung von öffentlich erreichbaren Diensten und dem Intranet.	Es erfolgte eine Trennung in DMZ und Intranet. Die DMZ enthält den online zur Verfügung gestellten Webserver. Im Intranet sind alle unternehmensinternen Server und Client-Geräte
Die netzwerkrelevanten internen Dienste DNS und DHCP sind auf einem separaten System bereitzustellen um Abhängigkeiten von der Firewall auszuschließen.	DNS und DHCP wurden auf einem Server im Intranet aufgesetzt und verteilen Konfigurationen an Geräte im Intranet, sowie bieten die Möglichkeit zur Namensauflösung
Doubtful-Joy setzt im Bereich der Server ausschließlich RedHat und binärkompatible Systeme ein. Alle OS-Installationen der Server folgen dieser System-Strategie.	Alle Geräte wurden mit CentOS 9 installiert.

Auf Grundlage des dynamischen Wachstums erwartet Doubtful-Joy eine begründete Empfehlung a) zur technischen Bereitstellung der IT-Infrastruktur | Upgrades sind jederzeit möglich (Hardware/Software), Upgrades sind schneller realisierbar (sofort) b) zum zukunftssicheren Systembetrieb der Lösung im Kontext von „make or buy“. | Die Wartung der Server ist einfacher und günstiger, da Fachkräfte vor Ort zur Verfügung stehen; |

Ergebnis und Zeitaufwand

- alle Anforderungen wurden erfüllt
- Zeitaufwand war per se nicht sehr hoch (3-4h für die Installation und Konfiguration aller VMs (Teil1) noch einmal 2h für den zweiten Teil)
- als endgültige Lösung für ein Unternehmen, würden jedoch noch einige Installationen und Konfigurationen fehlen

Optimierungen

- Domain Controller (entweder Windows AD-Server, oder freeipa (linuxbasiert)) für die zentrale Authentifizierung und Autorisierung, sowie Freigabe von Ordnern für berechtigte Personen (Berechtigungskonzept)
- auf dem IPFire sollte nicht nur Destination Natting, sondern auch Source Natting betrieben werden
  - Source Natting damit die eigentliche Server-Adresse des Webservers den Nutzern nicht bekannt gemacht wird -> Es wird dann so angezeigt, dass die Firewall selbst den Dienst zur Verfügung stellt
- regelmäßiges Datenbankbackup für den Datenbankserver, welches jedoch auf einem zusätzlichen Server gelagert wird
- einen Mailserver, welcher zusätzlich abgesichert werden kann

3.2 Analyse

Home

Aufgabe 1

Begründen Sie die öffentliche Erreichbarkeit/Nichterreichbarkeit der Systeme

Firewall-System öffentlich erreichbar: **Nein**, weil ein externer Zugriff auf die Firewall zu einer Sicherheitslücke führt, durch die Einstellungen an der Firewall vorgenommen werden können. Somit würden Externe Zugriff auf das Netzwerk erhalten.

DNS öffentlich erreichbar: **Nein**, da interne DNS-Server erfordern keine Authentifizierung. Somit kann meist jeder, der sich im Netzwerk befindet, interne DNS-Anfragen tätigen. Diese DNS-Server speichern alle Servernamen und IP-Adressen für private Domains, welche durch Angreifer ausgespäht und verfälscht werden könnten.

DHCP öffentlich erreichbar: **Nein**, da durch externe Angriffe sonst keine Clients mehr in das Netzwerk eingebunden werden können. Ebenso könnten IP-Adressen ausgespäht und letztendlich geändert werden.

Web-Server öffentlich erreichbar: **Ja**, da dieser aus dem Internet mithilfe von HTTPS über Port 443 aufrufbar sein sollte, damit Mandaten darauf zugreifen und Tickets erstellen können.

Datenbank-Server öffentlich erreichbar: **Nein**, da dieser sensible Daten wie Anmeldedaten der Website-Nutzer enthält, welche geschützt werden müssen.

[Pi-Hole] öffentlich erreichbar: **Nein**, da Angreifer somit Einstellungen tätigen können. Hierbei kann es sich um Änderungen von gefilterten Webseiten

handeln, um die internen Nutzer auf diese weiterzuleiten. Darüber hinaus werden DNS-Anfragen von der Software Pi-Hole gespeichert und dient somit als eigener DNS-Server, welcher ohne starke Absicherung nicht öffentlich erreichbar gemacht werden sollte.

[Mailproxy für eingehende Mails] öffentlich erreichbar: **Ja**, da sonst keine Mails in das interne Netzwerk weitergeleitet werden können.

[Existierender Mailserver] öffentlich erreichbar: Nein, da eingehende Mails vom Mailproxy an diesen weitergeleitet und anschließend verteilt werden. Wenn der Mailserver von extern erreichbar wäre, könnte dieser mit Schadmail belagert und darüber hinaus auch die Datenpakete ausspioniert werden. ## Aufgabe 2

Beschreiben Sie die Akteure und den jeweiligen Kommunikationsweg über die Zwischensysteme zu den IT-Endsystemen für folgenden Anwendungsfälle:

a) Ticket erstellen und in DB speichern Der Kunde greift vom Host-Rechner mit Hilfe eines Browsers auf den Web-Server zu, welcher aus dem Internet abrufbar ist und erstellt ein Ticket. Dabei läuft die Kommunikation über die IP-Adresse des Gateways des VMNet8, passiert die Firewall des IP-Fires, wird über das Gateway in das VMnet2. Die Daten des Tickets werden auf den Datenbank-Server übertragen und eingepflegt. Hierzu muss aus dem VMNet2 auf das VMNet1 über die Firewall des IPFires zugegriffen werden. Der Kunde kann jedoch auch telefonisch Kontakt mit dem Kundenservice herstellen. Der Mitarbeiter erzeugt anschließend über das interne VMNet1 auf den Webserver im VMNet2 ein Ticket, welches wieder in die Datenbank, welche im VMNet1 liegt, eingepflegt wird.

b) Administration von FW, DNS- und DHCP-Server Die Administration der Dienste erfolgt aus dem internen VMNet1-Netz durch einen Mitarbeiter mit administrativen Rechten. Ausgehend vom Administrator-Rechner kann eine ssh-Verbindung mit dem DNS- und DHCP-Server im Netz hergestellt werden, sofern der Port 22 auf dem Server geöffnet ist. Die Firewall kann über die Weboberfläche des IPFire vom Administrator-Rechner aus konfiguriert werden. Hierzu werden die IP-Adresse und der Port :444 im Browser eingegeben.

c) Administration des Web-Servers Für die Konfiguration des Web-Servers wird aus dem VMNet1 auf den Web-Server im VMNet2 über die Firewall des IPFires und des Web-Servers zugegriffen. Hierzu wird durch einen Administrator eine ssh-Verbindung hergestellt. Zu beachten ist, dass die Installationen eine Internetverbindung benötigen, welche über den Kinder- und Jugendschutzproxy läuft.

d) Datenbankabfragen zur Supportsteuerung (z.B. Anzahl offener Tickets) Die Datenbankabfragen können vom Administrator über eine ssh-Verbindung auf dem Datenbankserver getätigt werden, sofern der Port 22 des Datenbankservers geöffnet ist.

e) [PI-Hole] Das PI-Hole wird als Werbeblocker eingesetzt. Somit wird aus dem internen VMNet1 eine Anfrage über die Firewall des IPFire und das VMNet8 ins Internet getätigt und somit eine Webseite aufgerufen. Der Kommunikationsweg läuft wie bei einer gewöhnlichen DNS-Anfrage.

f) [Mailkommunikation] Die Mailkommunikation erfolgt via Client-Server-Prinzip. Der Client schickt seine Mail an den SMTP-Server (entweder im eigenen Subnetz, oder über ein Gateway an ein anderes Subnetz). Der Server kommuniziert mit dem DNS-Server, um den Empfänger der Mail zu ermitteln. Anschließend leitet der DNS-Server die Informationen an den SMTP-Server zurück, welcher letztendlich die Nachricht an den Empfänger übermittelt.