ネットワークサービスの設定

情報学部コンピュータ科学科 3 年 野原健汰 (学籍番号:101730279)
メールアドレス:nohara.kenta@e.mbox.nagoya-u.ac.jp

2019 年 5 月 16 日、5 月 23 日実施

目次

1 概要

本実験では、ルータ及び WWW サーバ、PC を Ethernet ケーブルを用いて接続することによりローカルネットワークを作成し、続いて DHCP サービスの設定、ファイアウォールの設定、WWW サービスの設定を行うことにより、外部と接続可能なネットワークを作成した。2 では実験全体の目的と実験機器、3〜7 では各実験の目的と概要、実験方法、実験結果、考察、8 では実験全体のまとめを記し、9〜13 では各実験に関連した調査課題を行った。

2 はじめに

2.1 実験の目的

本実験では、ローカルネットワークの設定及び DHCP サービスの設定、ファイアウォールの設定、WWW サービスの設定を実験的に検証し、ネットワークの仕組みの基礎と各設定の方法及び設定の意義、コマンドの出力結果を理解し、ネットワークに関する知識を身につける。そこで本レポートの前半では、初期環境の設定、ネットワークの設定、DHCP サービスの設定、ファイアウォールの設定、WWW サービスの設定を行うことでネットワークの仕組みと各設定の方法及びその設定の意義、コマンドの出力結果を理解する (課題 1〜5)。本レポートの後半では、TCP/IP 通信のパケットのヘッダ情報及びセキュリティ強化の動向などを調査することによりネットワークに関する理解を深める (調査課題 1〜5)。

2.2 実験機器

2.2.1 ルータ

コンピュータネットワークの中継・転送機器でありデータの転送経路を選択・制御する機能を持つ。
本実験では machine1 がルータである。

2.2.2 WWW サーバ用 PC

一般的に用いられる PC と同じ技術・仕様で設計。製造されたサーバコンピュータ
本実験では machine2 が WWW サーバ用 PC である。

2.2.3 PC

本実験では machine3 が PC である。

2.2.4 PC 切替器

キーボードやマウス、モニタなどの 1 つの周辺機器を複数のデバイスで使用できるようにしたもの。

2.2.5 モニタ

2.2.6 USB キーボード/USB マウス

2.2.7 Ethernet ストレートケーブル 1 本

LAN を構成する機器間を接続する通信ケーブル。クロスケーブルと異なりケーブル内の銅線が途中で交差しておらず、両端のピン配列は同じである。PC とスイッチとを接続する場合などはストレートケーブルを使用する。

2.2.8 Ethernet クロスケーブル 2 本

LAN を構成する機器間を接続する通信ケーブル。ストレートケーブルと異なりケーブル内の銅線が途中で交差しており、両端のピン配列は異なる。PC と PC を接続する場合はクロスケーブルを使用する。

3 [課題 1] 初期環境の設定

3.1 目的と概要

本実験では、fdisk、df、lvdisplay 等のコマンドを実行し、その出力結果を確認することで各コマンドの意味、出力結果の情報を理解する。

3.2 実験方法

3.2.1 環境の設定

ルータ (以後 machine1)、WWW サーバ (以後　machine 2)、PC(以後 machine 3) を PC 切替器を介してキーボード及びマウスと接続した。キーボードは PC 切替器のキーボード用端子に、マウスは PC 切替器のマウス用接続端子に接続した。PC 切替器と machine 1 及び machine 2、machine 3 は RGB ケーブルと USB ケーブルを用いて接続した。PC 切替器とモニタを RGB ケーブルを用いて接続した。Ethernet クロスケーブルを用いて machine 1 と machine 2、machine 1 と machine 3 の LAN インターフェース間を直接接続した。machine 1 と上位ネットワークは HUB を介して接続した。

以下の実験は machine 1 及び machine 2 でそれぞれ行った。

3.2.2 Linux カーネルリリース番号の確認

以下のコマンドを実行し、現在動作している Linux カーネルを確認した。

```
uname -r
```

3.2.3 ファイルシステムの確認

以下のコマンドを実行し、ファイルシステムのマウントポイントを記述した設定ファイル/etc/fstab の内容を確認した。

```
cat /etc/fstab
```

以下のコマンドを実行し、ディスクパーティションとマウントされているファイルシステムを確認した。

```
fdisk -l
df
```

以下のコマンドを実行し、論理ボリュームの内容を確認した。

```
lvdisplay
```

3.2.4 ホスト名の設定

- machine 1

  以下のコマンドを実行し、ホスト名を設定した。

  ```
  hostname icesc16.ice.nuie.nagoya-u.ac.jp
  ```

  vi エディタで設定ファイル/etc/hostname の内容を以下のように変更し、ホスト名の恒久的変更を設定した。

  ```
  #vi /etc/hostname
  127.0.0.1 icesc16.ice.nuie.nagoya-u.ac.jp localhost.localdomain localhost
  ```

- machine 2

  以下のコマンドを実行し、ホスト名を設定した。

  ```
  hostname www6.ice.nuie.nagoya-u.ac.jp
  ```

  vi エディタで設定ファイル/etc/hostname の内容を以下のように変更し、ホスト名の恒久的変更を設定した。

  ```
  #vi /etc/hostname
  127.0.0.1 www6.ice.nuie.nagoya-u.ac.jp localhost.localdomain localhost
  ```

3.2.5 SELinux の状態確認と無効化

- 現在の設定状況の確認
  以下のコマンドを実行し、現在の設定状況を確認した。

```
cat /etc/sysconfig/selinux
getenforce
```

- SELinux の設定を permissive モードに変更
  以下のコマンドを実行し、SELinux の設定を permissive モードに変更した。

```
setenforce 0
```

  変更したら、SELinux の設定ファイル /etc/sysconfig/selinux の内容を vi エディタで以下のように変更し、SELinux の設定を恒久的にした。

```
SELINUX=permissive
```

3.3 実験結果

3.3.1 Linux カーネルリリース番号の確認の結果

- machine1

```
[root@localhost ~]# uname -r
3.10.0-862.el7.x86_64
```

- machine2

```
[root@localhost ~]# uname -r
3.10.0-862.el7.x86_64
```

3.3.2 ファイルシステムのマウントポイントを記述した設定ファイル/etc/fstab の内容を確認した結果

- machine1

```
[root@localhost ~]# cat /etc/fstab

#
#/etc/fstab
#Created by anaconda on Sat Apr 6 00:42:18 2019
#
#Accessible filesystems, by reference, are maintained under '/dev/disk'
#See man pages fstab(5), findfs(8), mount(8) and or blkid(8) for more info
#
/dev/mapper/centos-root / xfs defaults 0 0
UUID=5eb200c5-354e-419c-9306-bdcc507c72c5 /boot xfs defaults 0 0
/dev/mapper/centos-home /home xfs defaults 0 0
/dev/mapper/centos-swap swap swap defaults 0 0
```

- machine2

```
[root@localhost ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Sat Apr  6 00:42:18 2019
#
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info
#
/dev/mapper/centos-root / xfs  defaults   0 0
UUID=5eb200c5-354e-419c-9306-bdcc507c72c5 /boot  xfs  defaults  0 0
/dev/mapper/centos-home /home xfs    defaults  0 0
/dev/mapper/centos-swap swap  swap  defaults    0 0
```

1 列目はデバイス名、2 列目はマウントポイント、3 列目はファイルシステムの種類、4 列目はマウント時のオプション、5 列目はファイルシステムを dump するかしないか (0 ならばしない 1 ならばする)、6 列目は起動時に fsck がチェックする順番 (1 はチェックの優先度が 1 番高い、2 はその他のファイルシステムを入れればよい、0 はファイルシステムはチェックされない) をそれぞれ表している。2 行目は linux ディスクの UUID を表している。UUID とは一意な識別子である。linux では windows とは異なり、ストレージデバイスを接続しただけではファイルの読み書きができない。また、linux は windows と異なり、パーティション単位でファイルのツリー構造を持つことができず、複数のパーティションも 1 つのツリー構造にまとめて扱う。デバイスにあるパーティションをこのツリー構造のどこかにディレクトリとして登録する作業がマウントであり、登録するディレクトリがマウントポイントである。(文献 [2] 参照)

3.3.3 ディスクパーティション確認の結果

- machine1

```
[root@localhost ~]# fdisk -l

Disk /dev/sda: 250.1 GB, 250059350016 bytes, 488397168 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk label type: dos
Disk identifier: 0x00016021

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048     2099199     1048576   83  Linux
/dev/sda2         2099200   488396799   243148800   8e  Linux LVM

Disk /dev/mapper/centos-root: 53.7 GB, 53687091200 bytes, 104857600 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/centos-swap: 8321 MB, 8321499136 bytes, 16252928 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

```
Disk /dev/mapper/centos-home: 187.0 GB, 186969489408 bytes, 365174784 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

fstab で確認したマウントされているデバイスのパーティション情報を表している。/dev/sda には 2 つのパーティションが作成されていることがわかる。/dev/sda2 は LVM と記されていることから論理ボリュームマネージャーであることがわかる。

- machine2

  script がとれていなかったため省略するが、machine1 と同じ結果であった。

3.3.4 マウントされているファイルシステム確認の結果

- machine1

```
[root@localhost ~]# df
Filesystem              1K-blocks    Used Available Use% Mounted on
/dev/mapper/centos-root  52403200 1081936  51321264   3% /
devtmpfs                  3936440       0   3936440   0% /dev
tmpfs                     3949112       0   3949112   0% /dev/shm
tmpfs                     3949112    8848   3940264   1% /run
tmpfs                     3949112       0   3949112   0% /sys/fs/cgroup
/dev/sda1                 1038336  145900    892436  15% /boot
/dev/mapper/centos-home 182498240   32944 182465296   1% /home
tmpfs                      789824       0    789824   0% /run/user/0
```

- machine2

```
[root@localhost ~]# df
Filesystem              1K-blocks    Used Available Use% Mounted on
/dev/mapper/centos-root  52403200 1007316  51395884   2% /
devtmpfs                  3913236       0   3913236   0% /dev
tmpfs                     3925932       0   3925932   0% /dev/shm
tmpfs                     3925932    9044   3916888   1% /run
tmpfs                     3925932       0   3925932   0% /sys/fs/cgroup
/dev/sda1                 1038336  146052    892284  15% /boot
/dev/mapper/centos-home 182498240   32944 182465296   1% /home
tmpfs                      785188       0    785188   0% /run/user/0
```

1 列目はデバイス名、2 列目は全ディスク容量、3 列目は使用容量、4 列目は空き容量、5 列目は使用率、6 列目はマウントポイントを表している。

3.3.5 論理ボリュームの内容確認の結果

- machine1

```
[root@localhost ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/centos/swap
  LV Name                swap
  VG Name                centos
```

```
  LV UUID                5S8Xtv-OZ7e-3krn-HqEs-8xHA-CYFl-BfapS4
  LV Write Access        read/write
  LV Creation host, time localhost, 2019-03-22 12:04:24 +0900
  LV Status              available
  # open                 2
  LV Size                7.75 GiB
  Current LE             1984
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:1

  --- Logical volume ---
  LV Path                /dev/centos/home
  LV Name                home
  VG Name                centos
  LV UUID                vanhId-NRrd-icZs-ugHk-PTLw-F5zJ-XgnuQb
  LV Write Access        read/write
  LV Creation host, time localhost, 2019-03-22 12:04:25 +0900
  LV Status              available
  # open                 1
  LV Size                <174.13 GiB
  Current LE             44577
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:2

  --- Logical volume ---
  LV Path                /dev/centos/root
  LV Name                root
  VG Name                centos
  LV UUID                zNAdo6-xqs9-t9PR-iKrf-C855-jD6d-Lv0l7q
  LV Write Access        read/write
  LV Creation host, time localhost, 2019-03-22 12:04:26 +0900
  LV Status              available
  # open                 1
  LV Size                50.00 GiB
  Current LE             12800
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

- machine2

```
[root@localhost ~]# lvdisplay
  --- Logical volume ---
  LV Path                /dev/centos/swap
  LV Name                swap
  VG Name                centos
  LV UUID                PohTZH-fMlu-Lf0h-Mdhh-Z2xs-I2QH-ZM8ush
  LV Write Access        read/write
  LV Creation host, time localhost, 2019-04-06 00:42:14 +0900
  LV Status              available
```

```
  # open                 2
  LV Size                7.75 GiB
  Current LE             1984
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:1

  --- Logical volume ---
  LV Path                /dev/centos/home
  LV Name                home
  VG Name                centos
  LV UUID                XYwPiS-jXOD-IKEh-Jzzo-NDvs-qfED-uV8ABr
  LV Write Access        read/write
  LV Creation host, time localhost, 2019-04-06 00:42:14 +0900
  LV Status              available
  # open                 1
  LV Size                <174.13 GiB
  Current LE             44577
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:2

  --- Logical volume ---
  LV Path                /dev/centos/root
  LV Name                root
  VG Name                centos
  LV UUID                Oz554A-DomL-OGcW-B3u7-yAir-UU6R-Fa3bE1
  LV Write Access        read/write
  LV Creation host, time localhost, 2019-04-06 00:42:16 +0900
  LV Status              available
  # open                 1
  LV Size                50.00 GiB
  Current LE             12800
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:0
```

swap、home、root という名前の 3 つの論理ボリュームが作成されている。論理ボリュームはボリュームグループ上に作成された仮想的なパーティションである。論理ボリュームを作成することで、物理ディスク及びそのパーティションをそのサイズに関わらず、単一のストレージ・ソースとして抽象化して把握できるようになる。詳しくは考察にて追及する。(文献 [3] 参照)

3.3.6 現在の設定状況の確認の結果

- machine1

```
[root@icesc16 ~]# cat /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
```

```
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

```
[root@icesc16 ~]# getenforce
Enforcing
```

- machine2

```
[root@www6 ~]# cat /etc/sysconfig/selinux

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

```
[root@www6 ~]# getenforce
Enforcing
```

セキュリティ関連の Linux カーネル制御機能である SELinux が有効 (Enforcing) になっていることが分かる。SELinux はデフォルトで有効になっている。

3.3.7 SELinux の設定を permissive モードに変更した結果

- machine1

```
[root@icesc16 ~]# cat /etc/sysconfig/selinux
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

- machine2

```
[root@www6 ~]# cat /etc/sysconfig/selinux
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three two values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

SELinux を permissive に変更して機能を無効化した。これによりアクセス制御は無効になる (警告メッセージは出る)。

3.4 考察

今回の実験では、/dev/sda/というデバイスに CentOS がインストールされて/dev/sda1/と/dev/sda2/という 2 つのパーティションが作成された。そのうち/dev/sda2/は論理ボリュームマネージャであり centos という名前のボリュームグループを作成し、さらに root、swap、home という 3 つの論理ボリュームに分割している。ここで、論理ボリュームを作成するメリットについて考察する。
論理ボリュームを作成するメリットは大きく 4 つあると考えられる。1 つ目は小さいストレージデバイスを合わせて巨大な 1 つのパーティションが作成できることである。小さいストレージのデバイスは単一では使い物にならないが、複数組み合わせて一つのボリュームにすることで大きなストレージのデバイスを作成できる。2 つ目は大きなストレージのデバイスを小さなストレージのデバイスに分割することができることである。ただ大きなストレージデバイスを小さなストレージデバイスに分割するだけなら論理ボリュームマネージャを使わなくても実現できるが、論理ボリュームマネージャを使うと細かく分割して管理し、パーティションが足りなくなったらサイズを拡大することができるという点である。3 つ目はスナップショット (その瞬間のボリュームのイメージのこと) を取得できることである。スナップショットはコピーバックアップとは異なり、基本的にデータは持っていない。スナップショット取得後にデータが更新された場合は更新される前のデータをスナップショットが保持することでスナップショット取得時の状態を保持する。スナップショットは更新される前のデータだけを持つことでボリュームのサイズを節約している。4 つ目はパーティションの設定や管理が容易になることである。例えばレンタルサーバーではサイズ契約を変更したい場合に容易にそのサイズを変更することができる。論理ボリュームマネージャを使わない場合は一度パーティションを作成すると簡単にサイズを変更できない。

4 [課題 2] ネットワーク設定

4.1 目的と概要

本実験では、nmcli コマンドを用いたネットワークインターフェースの接続設定を実験的に検証し、接続設定方法を確認、コマンド結果の意味を理解する。

4.2 実験方法

以下は machine 1 及び machine 2 でそれぞれ行った。

4.2.1 現在の状況の確認

- ネットワークデバイスの確認以下のコマンドを実行し、ネットワークデバイスの確認を行った。

```
nmcli device status
nmcli device show
```

- ネットワーク接続の確認以下のコマンドを実行し、ネットワーク接続の確認を行った。

```
nmcli connection show –active
```

4.2.2 接続設定

- machine 1

  以下のコマンドを実行し、OS 起動時の自動接続指定を行った。

```
nmcli c m enp1s0 connection.autoconnect yes
nmcli c m enp2s0 connection.autoconnect yes
nmcli c m enp3s0 connection.autoconnect yes
```

  以下のコマンドをそれぞれ実行し、IP アドレスの設定を行った。

```
nmcli c m enp1s0 ipv4.method manual ipv4.addresses “192.168.100.16/24”
nmcli c m enp2s0 ipv4.method manual ipv4.addresses “192.168.150.1/24”
nmcli c m enp3s0 ipv4.method manual ipv4.addresses “192.168.200.1/24”
```

  以下のコマンドを実行し、デフォルトゲートウェイの設定を行った (LAN1 のみ)。

```
nmcli c m enp1s0 gateway “192.168.100.1”
```

  以下のコマンドを実行し、DNS サーバーの設定を行った。

```
nmcli c m enp1s0 ipv4.dns “10.10.1.2”
```

- machine 2

  以下のコマンドを実行し、OS 起動時の自動接続指定を行った。

```
nmcli c m enp3s0 connection.autoconnect yes
```

4.2.3 ネットワーク接続の有効化

以下のコマンドをそれぞれ実行し、ネットワーク接続を有効化した。

- machine 1

```
nmcli con down enp1s0
nmcli con up enp1s0
nmcli con down enp2s0
nmcli con up enp2s0
nmcli con down enp3s0
nmcli con up enp3s0
```

- machine 2

```
nmcli con down enp3s0
nmcli con up enp3s0
```

4.2.4 各種情報の確認

以下のコマンドをそれぞれ実行し、各デバイスの設定の確認、ルーティング情報の確認を行った。

```
ip addr show
ip route
```

4.3 実験結果

4.3.1 ネットワークデバイスの確認の結果

- machine 1

```
[root@icesc16 ~]# nmcli device status

DEVICE  TYPE      STATE         CONNECTION
enp1s0  ethernet  disconnected  --
enp2s0  ethernet  disconnected  --
enp3s0  ethernet  disconnected  --
enp4s0  ethernet  unavailable   --
lo      loopback  unmanaged     --
```

1 列目はデバイス名、2 列目はコネクションタイプ、3 列目は接続状態、4 列目は接続を表している。machine1 では enp1s0、enp2s0、enp3s0 の 3 種類のネットワークデバイスを使用できる (この段階では接続されていない) ことがわかる。lo はローカルループバックを表している。

```
[root@icesc16 ~]# nmcli device show

GENERAL.DEVICE:                         enp1s0
GENERAL.TYPE:                           ethernet
```

```
GENERAL.HWADDR:                         00:E0:67:12:2D:B4
GENERAL.MTU:                            1500
GENERAL.STATE:                          30 (disconnected)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
WIRED-PROPERTIES.CARRIER:               on

GENERAL.DEVICE:                         enp2s0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         00:E0:67:12:2D:B5
GENERAL.MTU:                            1500
GENERAL.STATE:                          30 (disconnected)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
WIRED-PROPERTIES.CARRIER:               on

GENERAL.DEVICE:                         enp3s0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         00:E0:67:12:2D:B6
GENERAL.MTU:                            1500
GENERAL.STATE:                          30 (disconnected)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
WIRED-PROPERTIES.CARRIER:               on

GENERAL.DEVICE:                         enp4s0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         00:E0:67:12:2D:B7
GENERAL.MTU:                            1500
GENERAL.STATE:                          20 (unavailable)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
WIRED-PROPERTIES.CARRIER:               off

GENERAL.DEVICE:                         lo
GENERAL.TYPE:                           loopback
GENERAL.HWADDR:                         00:00:00:00:00:00
GENERAL.MTU:                            65536
GENERAL.STATE:                          10 (unmanaged)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
IP4.ADDRESS[1]:                         127.0.0.1/8
IP4.GATEWAY:                            --
IP6.ADDRESS[1]:                         ::1/128
IP6.GATEWAY:                            --
```

nmcli device status コマンドで確認したネットワークデバイスの詳細が表示されている。GENERAL.DEVICE はデバイス名、GENERAL.TYPE はコネクションタイプ、HWADDR は MAC アドレス、GENERAL.MTU は一度に送信できる最大のデータ量 (単位は byte)、GENERAL.STATE は接続状態をそれぞれ表している。ローカルループバックアドレスは自分自身を表す IP アドレスであり一般的に 127.0.0.1 が利用される。

- machine 2

```
[root@www6 ~]# nmcli device status
```

```
DEVICE  TYPE      STATE         CONNECTION
enp3s0  ethernet  disconnected  --
lo      loopback  unmanaged     --
wlp2s0  wifi      unmanaged     --
```

コマンドの見方は machine1 で示したため省略する。

machine2 では有線 LAN である enp3s0 のみ使用できる (この段階では接続されていない) ことがわかる。wlp2s0 は無線 LAN を表している。

```
[root@www6 ~]# nmcli device show

GENERAL.DEVICE:                         enp3s0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         94:C6:91:A8:C9:54
GENERAL.MTU:                            1500
GENERAL.STATE:                          30 (disconnected)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
WIRED-PROPERTIES.CARRIER:               on

GENERAL.DEVICE:                         lo
GENERAL.TYPE:                           loopback
GENERAL.HWADDR:                         00:00:00:00:00:00
GENERAL.MTU:                            65536
GENERAL.STATE:                          10 (unmanaged)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
IP4.ADDRESS[1]:                         127.0.0.1/8
IP4.GATEWAY:                            --
IP6.ADDRESS[1]:                         ::1/128
IP6.GATEWAY:                            --

GENERAL.DEVICE:                         wlp2s0
GENERAL.TYPE:                           wifi
GENERAL.HWADDR:                         DC:8B:28:55:19:23
GENERAL.MTU:                            1500
GENERAL.STATE:                          10 (unmanaged)
GENERAL.CONNECTION:                     --
GENERAL.CON-PATH:                       --
IP4.GATEWAY:                            --
IP6.GATEWAY:                            --
```

コマンドの見方は machine1 で示したため省略する。

4.3.2 ネットワーク接続の確認の結果

- machine 1

```
[root@icesc16 ~]# nmcli connection show --active
NAME  UUID  TYPE  DEVICE
```

- machine 2

```
[root@www6 ~]# nmcli connection show -active
NAME  UUID  TYPE  DEVICE
```

この段階では、ネットワーク接続されているデバイスが存在しないため何も表示されなかった。

4.3.3 ネットワーク接続の有効化の結果

- machine 1

```
[root@icesc16 ~]# nmcli con down enp1s0
Connection 'enp1s0' successfully deactivated
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/11)
[root@icesc16 ~]# nmcli con up enp1s0
Connection successfully activated
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/14)
[root@icesc16 ~]# nmcli con down enp2s0
Connection 'enp2s0' successfully deactivated
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/17)
[root@icesc16 ~]# nmcli con up enp2s0
Connection successfully activated
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/52)
[root@icesc16 ~]# nmcli con down enp3s0
Connection 'enp3s0' successfully deactivated
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/20)
[root@icesc16 ~]# nmcli con up enp3s0
Connection successfully activated
(D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/56)
```

- machine 2

```
[root@www6 ~]# nmcli con down enp3s0
Connection 'enp3s0' successfully deactivated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/2)
[root@www6 ~]# nmcli con up enp3s0
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/5)
```

一度 down して up することで有効化される。

4.3.4 各種情報の確認

- machine 1

```
[root@icesc16 ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:e0:67:12:2d:b4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.16/24 brd 192.168.100.255 scope global noprefixroute enp1s0
       valid_lft forever preferred_lft forever
    inet6 fe80::1a08:bb64:b6fa:6306/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
```

```
    link/ether 00:e0:67:12:2d:b5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.150.1/24 brd 192.168.150.255 scope global noprefixroute enp2s0
       valid_lft forever preferred_lft forever
    inet6 fe80::f6cc:73c7:7638:c48c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:e0:67:12:2d:b6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.200.1/24 brd 192.168.200.255 scope global noprefixroute enp3s0
       valid_lft forever preferred_lft forever
    inet6 fe80::894a:645e:4f45:794d/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
5: enp4s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast state DOWN
group default qlen 1000
    link/ether 00:e0:67:12:2d:b7 brd ff:ff:ff:ff:ff:ff
```

```
[root@icesc16 ~]# ip route
default via 192.168.100.1 dev enp1s0 proto static metric 100
192.168.100.0/24 dev enp1s0 proto kernel scope link src 192.168.100.16 metric 100
192.168.150.0/24 dev enp2s0 proto kernel scope link src 192.168.150.1 metric 103
192.168.200.0/24 dev enp3s0 proto kernel scope link src 192.168.200.1 metric 104
```

- machine 2

```
[root@www6 ~]# ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
 group default qlen 1000
    link/ether 94:c6:91:a8:c9:54 brd ff:ff:ff:ff:ff:ff
    inet 192.168.150.100/24 brd 192.168.150.255 scope global noprefixroute dynamic
enp3s0
       valid_lft 3435sec preferred_lft 3435sec
    inet6 fe80::15b7:4ca2:a0f3:9ba3/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: wlp2s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
group default qlen 1000
    link/ether dc:8b:28:55:19:23 brd ff:ff:ff:ff:ff:ff
```

```
[root@www6 ~]# ip route
default via 192.168.150.1 dev enp3s0 proto dhcp metric 100
192.168.150.0/24 dev enp3s0 proto kernel scope link src 192.168.150.100 metric 100
```

以下に ip addr コマンドの見方を示す。

表 1　ip addr コマンドの意味

lo、enp1s0、enp2s0、enp3s0、enp4s0、wlp2s0	デバイス名 (lo はローカルループバック)
LOOPBACK、BROADCAST、 MULTICAST、UP、LOWER_UP	LOOPBACK... ループバック対応 BROADCAST... ブロードキャスト対応 MULTICAST... マルチキャスト対応 UP... ネットワークインタフェースが有効 LOWER_UP... デバイスにケーブルが繋がっている (有線 LAN)
mtu	一度に送信できるデータの最大値 (byte) (Linux 標準値 1500)
qdisc	ネットワークへの送信を待つデータの保存規則 pfifo_fast... パケットの優先度を考慮した qdisc(Linux デフォルト)
state	UP... ネットワークインターフェースが作動中 DOWN... ネットワークインターフェースが作動していない
group default	グループインターフェース
qlen	queue の長さ
link/loopback、link/either	MAC アドレス
brd ff:ff:ff:ff:ff:ff	ブロードキャストアドレス
inet、inet6	inet...IPv4 アドレス inet6...IPv6 アドレス
brd xxx.xxx.xxx.255	ディレクティッドブロードキャストアドレス
scope	送信先指定 global... 他ネットワークへのゲートウェイを 経由した unicast 通信による経路 link... 自身が属するネットワーク
valid_lft、preferred_lft	valid_lft... 有効な IPv4 アドレスの有効期限 preferred_lft... 適切な IPv4 アドレスの有効期限

以下に ip route コマンドの見方を示す。

表 2　ip route コマンドの意味

via	ネクストホップのルーター
dev	対象デバイス名
proto	kernel... カーネルが自動生成した経路 dhcp...DHCP が自動生成した経路
metric	ネットワーク間の仮想的な距離
scope	送信先を指定 link... 自身が属するネットワーク
src	送信元を指定

4.4　考察

aaaaaaaaaaaaaaaaaaaaa

5　[課題 3]DHCP サービスの設定

5.1　目的と概要

本実験では、yum コマンドを用いた DHCP サーバのインストールと設定ファイルの修正を行い、DHCP サーバを動作させることにより、DHCP の動作方法を確認し理解する。

5.2　実験方法

以下の実験は machine 1 で行った。

5.2.1　稼働中のサービスの確認

以下のコマンドを実行し、稼働中のサービスの確認を行った。

```
systemctl list-units –type=service
```

5.2.2　firewall の停止

以下のコマンドをそれぞれ実行し、firewall を一時的に停止した。

```
systemctl stop firewalld
systemctl status firewalld
```

5.2.3　DHCP サービスのインストールと設定

- DHCP 関連パッケージの確認と DHCP サーバのインストール
  以下のコマンドを実行し、DHCP 関連パッケージの確認と DHCP サーバのインストールを行った。

```
yum list dhcp-
yum info dhcp
yum install dhcp
```

- 設定ファイルの修正
  設定ファイル (/etc/dhcp/dhcpd.conf) を以下のように修正した。ゲートウェイ及びネットマスク、DNS サーバを指定した。
  リース範囲は 192.168.150.100〜192.168.150.250 及び 192.168.200.100〜192.168.200.250 とした。
  リース期間はデフォルト 1 時間 (3600 秒)、最大 1 日 (86400 秒)、machine2 のリース期間は 30 日 (2592000 秒) に設定した。

```
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.example
#   see dhcpd.conf(5) man page
#

default-lease-time 3600;
max-lease-time 86400;
option domain-name-servers 10.10.1.2;

subnet 192.168.150.0 netmask 255.255.255.0 {
    range 192.168.150.100 192.168.150.250;
    option routers 192.168.150.1;
}

subnet 192.168.200.0 netmask 255.255.255.0 {
    range 192.168.200.100 192.168.200.250;
    option routers 192.168.200.1;
}

host www6 {
    hardware ethernet 94:c6:91:a8:c9:54;
    fixed-address 192.168.150.2;
    default-lease-time 2592000;
}
```

- DHCP サーバの再起動以下のコマンドを実行し、DHCP サーバの再起動を行った。

```
systemctl restart dhcpd
```

- DHCP サーバのブート時自動起動の設定以下のコマンドを実行し、DHCP サーバのブート時自動起動を設定した。

```
systemctl enable dhcpd
```

5.3 実験結果

5.3.1 稼働中のサービスの確認の結果

```
[root@icesc16 ~]# systemctl list-units --type=service

UNIT                                            LOAD   ACTIVE SUB     DESCRIPTION
auditd.service
loaded active running Security Auditing Service
crond.service
loaded active running Command Scheduler
dbus.service
loaded active running D-Bus System Message Bus
firewalld.service
loaded active running firewalld - dynamic firewall daemon
getty@tty1.service
loaded active running Getty on tty1
irqbalance.service
loaded active running irqbalance daemon
kdump.service
loaded active exited  Crash recovery kernel arming
kmod-static-nodes.service
loaded active exited  Create list of required static device nodes for the curren
lvm2-lvmetad.service
loaded active running LVM2 metadata daemon
lvm2-monitor.service
loaded active exited  Monitoring of LVM2 mirrors, snapshots etc. using dmeventd
lvm2-pvscan@8:2.service
loaded active exited  LVM2 PV scan on device 8:2
network.service
loaded active exited  LSB: Bring up/down networking
NetworkManager-wait-online.service
loaded active exited  Network Manager Wait Online
NetworkManager.service
loaded active running Network Manager
polkit.service
loaded active running Authorization Manager
postfix.service
loaded active running Postfix Mail Transport Agent
rhel-dmesg.service
loaded active exited  Dump dmesg to /var/log/dmesg
rhel-domainname.service
loaded active exited  Read and set NIS domainname from /etc/sysconfig/network
rhel-import-state.service
loaded active exited  Import network configuration from initramfs
rhel-readonly.service
loaded active exited  Configure read-only root support
rsyslog.service
loaded active running System Logging Service
sshd.service
loaded active running OpenSSH server daemon
systemd-backlight@backlight:acpi_video0.service
loaded active exited  Load/Save Screen Backlight Brightness of backlight:acpi_vi
systemd-journal-flush.service
loaded active exited  Flush Journal to Persistent Storage
systemd-journald.service
loaded active running Journal Service
systemd-logind.service
loaded active running Login Service
systemd-random-seed.service
loaded active exited  Load/Save Random Seed
```

```
systemd-readahead-collect.service
loaded active exited  Collect Read-Ahead Data
systemd-readahead-replay.service
loaded active exited  Replay Read-Ahead Data
systemd-remount-fs.service
loaded active exited  Remount Root and Kernel File Systems
systemd-sysctl.service
loaded active exited  Apply Kernel Variables
systemd-tmpfiles-setup-dev.service
loaded active exited  Create Static Device Nodes in /dev
systemd-tmpfiles-setup.service
loaded active exited  Create Volatile Files and Directories
systemd-udev-trigger.service
loaded active exited  udev Coldplug all Devices
systemd-udevd.service
loaded active running udev Kernel Device Manager
systemd-update-utmp.service
loaded active exited  Update UTMP about System Boot/Shutdown
systemd-user-sessions.service
loaded active exited  Permit User Sessions
systemd-vconsole-setup.service
loaded active exited  Setup Virtual Console
tuned.service
loaded active running Dynamic System Tuning Daemon
```

UNIT はサービス名、LOAD はユニット定義が正しくロードされたか、ACTIVE は高レベルのユニット起動状態、SUB は低レベルのユニット起動状態 (値はユニットの種類により異なる)、DESCRIPTION はサービスの詳細を表している。上から 4 つめのサービスで、firewall が稼働していることがわかる。

5.3.2 firewall 停止の確認結果

```
[root@icesc16 ~]# systemctl status firewalld●
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service;
enabled; vendor preset: enabled)
   Active: inactive (dead) since Thu 2019-05-16 15:49:46 JST; 9s ago
     Docs: man:firewalld(1)
  Process: 701 ExecStart=/usr/sbin/firewalld --nofork --nopid
$FIREWALLD_ARGS (code=exited, status=0/SUCCESS)
 Main PID: 701 (code=exited, status=0/SUCCESS)

May 16 13:50:26 localhost.localdomain systemd[1]:
Starting firewalld - dynamic firewall daemon...
May 16 13:50:27 localhost.localdomain systemd[1]:
Started firewalld - dynamic firewall daemon.
May 16 15:49:44 icesc16.ice.nuie.nagoya-u.ac.jp systemd[1]:
Stopping firewalld - dynamic firewall daemon...
May 16 15:49:46 icesc16.ice.nuie.nagoya-u.ac.jp systemd[1]:
Stopped firewalld - dynamic firewall daemon.
```

Active: inactive(dead) と表示されているため、firewall サービスが停止したことが確認できた。

5.3.3 DHCP 関連パッケージの確認結果

```
[root@icesc16 ~]# yum list dhcp-\*
```

```
Installed Packages
dhcp-common.x86_64                                      12:4.2.5-68.el7.centos
@anaconda
dhcp-libs.x86_64                                        12:4.2.5-68.el7.centos
@anaconda
Available Packages
dhcp.x86_64                                             12:4.2.5-68.el7.centos.1
base
dhcp-common.x86_64                                      12:4.2.5-68.el7.centos.1
base
dhcp-devel.i686                                         12:4.2.5-68.el7.centos.1
base
dhcp-devel.x86_64                                       12:4.2.5-68.el7.centos.1
base
dhcp-libs.i686                                          12:4.2.5-68.el7.centos.1
base
dhcp-libs.x86_64                                        12:4.2.5-68.el7.centos.1
base
```

インストール可能なパッケージ情報が表示されている。

```
[root@icesc16 ~]# yum info dhcp
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirrors.cat.net
 * extras: mirrors.cat.net
 * updates: ftp.jaist.ac.jp
Available Packages
Name        : dhcp
Arch        : x86_64
Epoch       : 12
Version     : 4.2.5
Release     : 68.el7.centos.1
Size        : 513 k
Repo        : base/7/x86_64
Summary     : Dynamic host configuration protocol software
URL         : http://isc.org/products/DHCP/
License     : ISC
Description : DHCP (Dynamic Host Configuration Protocol) is a protocol which allows
            : individual devices on an IP network to get their own network
            : configuration information (IP address, subnetmask, broadcast address,
            : etc.) from a DHCP server. The overall purpose of DHCP is to make it
            : easier to administer a large network.
            :
            : To use DHCP on your network, install a DHCP service (or relay agent),
            : and on clients run a DHCP client daemon.  The dhcp package provides
            : the ISC DHCP service and relay agent.
```

dhcp のパッケージの詳細情報 (サイズ、説明など) が表示されている。

5.4 考察

aaaaaaaaaaaaaaaaaaaaa

6 [課題 4] ファイアウォールの設定

6.1 目的と概要

本実験では、パケットフィルタリングについて iptables の設定を通して実験的に検証することにより、iptables の設定方法及びパケットフィルタリングの基礎知識を身につける。

6.2 実験方法

machine1 について下記実験を行った。

6.2.1 パケット転送の有効化

設定ファイル /etc/sysctl.d/10-ipv4.conf を作成し、以下を追加した。

```
net.ipv4.ip_forward=1
```

設定ファイルを変更したら、以下のコマンドで有効化した。
これによりパケット転送が有効化された。

```
sysctl –system
reboot
```

6.2.2 firewalld から iptables-services への変更

以下のコマンドを実行し、Netfilter のフロントエンドを firewalld から iptables に変更した。

```
systemctl stop firewalld
systemctl disable firewalld
yum install iptables-services
systemctl start iptables
systemctl status iptables
systemctl enable iptables
```

6.2.3 サンプルの実行

サンプルファイルを以下のコマンドで machine1 にコピーして実行した。

```
scp bv0572197@ssh.ice.nuie.nagoya-u.ac.jp:/pub1/jikken/cs-net/iptables-sample.sh
sh iptables-sample.sh
```

6.2.4 iptables の設定状況の確認

以下のコマンドを実行し、iptables の設定状況を確認した。

```
iptables -L -n
```

以下のコマンドを実行し、現在のルールを確認した。

```
iptables-save
```

6.2.5 設定スクリプトの修正

外部のパソコンで以下のようなファイルを作り、scp コマンドで machine1 にコピーした (iptables-sample2.sh)。
設定の詳細はコメントアウトで示し、ポート番号及び各オプションの意味等は実験結果にて示す。

```
#!/bin/sh

PATH=/sbin:/bin:/usr/bin:/usr/sbin

## 変数の定義
EXTERNAL_INTERFACE="enp1s0"        # 外側インタフェースの名前
DMZ_INTERFACE="enp2s0"             # DMZ インタフェースの名前
INTERNAL_INTERFACE="enp3s0"        # 内側インタフェースの名前

# 外側インタフェースの IP アドレス
IPADDR=‘ip addr show $EXTERNAL_INTERFACE | \
 sed -e ’s/^.*inet \([^ \/]*\).*$/\1/p’ -e d‘
# 内部ネットワーク・アドレス
INTERNAL_LAN=‘ip addr show $INTERNAL_INTERFACE | \
 sed -e ’s/^.*inet \([^ ]*\).*$/\1/p’ -e d‘

# DMZ ネットワーク・アドレス
DMZ_LAN=‘ip addr show $DMZ_INTERFACE | \
 sed -e ’s/^.*inet \([^ ]*\).*$/\1/p’ -e d‘

ANYWHERE="0.0.0.0/0"

## 以下の設定を実行している間はパケットの転送を停止する
echo 0 > /proc/sys/net/ipv4/ip_forward
```

```
## すでに設定されているルールを消去する
iptables -F
iptables -F -t nat

## ポリシーの初期設定 -> match しない場合の扱い
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

## ループバック・インタフェースの入出力を許可する
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

################################################################################
##
## INPUT チェーンの設定（デフォルト拒否）
##

iptables -A INPUT -i $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
    --dport 22 -j ACCEPT
iptables -A INPUT -i $INTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
    --dport 22 -j ACCEPT
iptables -A INPUT -i $INTERNAL_INTERFACE -p udp -m state --state NEW -m udp \
    --dport 67 -j ACCEPT
iptables -A INPUT -i $INTERNAL_INTERFACE -p icmp -j ACCEPT
iptables -A INPUT -i $DMZ_INTERFACE -p udp -m state --state NEW -m udp \
    --dport 67 -j ACCEPT
iptables -A INPUT -i $DMZ_INTERFACE -p icmp -j ACCEPT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT


################################################################################
##
## OUTPUT チェーンの設定（デフォルト拒否）
##

iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
    --dport 22 -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
```

```
    --dport 80 -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p tcp -m state --state NEW -m tcp \
    --dport 443 -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p udp -m state --state NEW -m udp \
    --dport 53 -j ACCEPT
iptables -A OUTPUT -o $EXTERNAL_INTERFACE -p icmp -j ACCEPT
iptables -A OUTPUT -o $INTERNAL_INTERFACE -p icmp -j ACCEPT
iptables -A OUTPUT -o $DMZ_INTERFACE -p tcp -m state --state NEW -m tcp \
    --dport 22 -j ACCEPT
iptables -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT


################################################################################
##
## FORWARD チェーンの設定（デフォルト拒否）
##
#NEW 新規パケット、ESTABLISHED 継続パケット、RELATED 関連パケットと呼ぶ

iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
 -m state --state NEW,ESTABLISHED -m tcp --dport 22 -j ACCEPT
#machine3 に入った新規 ssh パケット、継続 ssh パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p udp \
-m state --state NEW,ESTABLISHED -m udp --dport 53 -j ACCEPT
#machine3 に入った新規 dns パケット、継続 dns パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
-m state --state NEW,ESTABLISHED -m tcp --dport 80 -j ACCEPT
#machine3 に入った新規 http パケット、継続 http パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
-m state --state NEW,ESTABLISHED -m tcp --dport 443 -j ACCEPT
#machine3 に入った新規 https パケット、継続 https パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p icmp \
-j ACCEPT
#machine3 から machine1 に対して ping を許可

iptables -A FORWARD -i $EXTERNAL_INTERFACE -o $DMZ_INTERFACE -p tcp \
-m state --state NEW,ESTABLISHED -m tcp --dport 80 -j ACCEPT
```

```
#machine1 に入った新規 http パケット、継続 http パケットを machine2 に対して転送を許可

iptables -A FORWARD -i $EXTERNAL_INTERFACE -o $DMZ_INTERFACE -p tcp \
-m state --state NEW,ESTABLISHED -m tcp --dport 443 -j ACCEPT
#machine1 に入った新規 https パケット、継続 https パケットを machine2 に対して転送を許可

iptables -A FORWARD -i $DMZ_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
-m state --state NEW,ESTABLISHED -m tcp --dport 22 -j ACCEPT
#machine2 に入った新規 ssh パケット、継続 ssh パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $DMZ_INTERFACE -o $EXTERNAL_INTERFACE -p udp \
-m state --state NEW,ESTABLISHED -m udp --dport 53 -j ACCEPT
#machine2 に入った新規 dns パケット、継続 dns パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $DMZ_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
-m state --state NEW,ESTABLISHED -m tcp --dport 80 -j ACCEPT
#machine2 に入った新規 http パケット、継続 http パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $DMZ_INTERFACE -o $EXTERNAL_INTERFACE -p tcp \
-m state --state NEW,ESTABLISHED -m tcp --dport 443 -j ACCEPT
#machine2 に入った新規 https パケット、継続 https パケットを machine1 に対して転送を許可

iptables -A FORWARD -i $DMZ_INTERFACE -o $EXTERNAL_INTERFACE -p icmp -j ACCEPT
#machine2 から machine1 への ping を許可

iptables -A FORWARD -i $INTERNAL_INTERFACE -o $EXTERNAL_INTERFACE -p \
icmp -j ACCEPT
#machine3 から machine1 への ping を許可

iptables -A FORWARD -i $INTERNAL_INTERFACE -o $DMZ_INTERFACE -m state \
--state NEW,ESTABLISHED -j ACCEPT
#machine3 に入った全ての新規パケットと継続パケットを machine2 に対して転送を許可

iptables -A FORWARD -i $DMZ_INTERFACE -o $INTERNAL_INTERFACE -m state \
--state RELATED,ESTABLISHED -j ACCEPT
#machine2 に入った全ての関連パケットと継続パケットを machine3 に対して転送を許可

iptables -A FORWARD -i $DMZ_INTERFACE -o $EXTERNAL_INTERFACE -m state \
--state RELATED,ESTABLISHED -j ACCEPT
#machine2 に入った全ての関連パケットと継続パケットを machine1 に対して転送を許可
```

```
iptables -A FORWARD -i $EXTERNAL_INTERFACE -o $DMZ_INTERFACE -m state \
--state RELATED,ESTABLISHED -j ACCEPT
#machine1 に入った全ての関連パケットと継続パケットを machine2 に対して転送を許可

iptables -A FORWARD -i $EXTERNAL_INTERFACE -o $INTERNAL_INTERFACE -m \
state --state RELATED,ESTABLISHED -j ACCEPT
#machine1 に入った全ての関連パケットと継続パケットを machine3 に対して転送を許可

################################################################################
##
## NAT の設定
##

iptables -A POSTROUTING -t nat -s $INTERNAL_LAN -o $EXTERNAL_INTERFACE -j SNAT \
    --to-source $IPADDR
#POSTROUTING チェーンで machine1 から出ていく machine3 の IP アドレスを
#machine1 の IP アドレスに変換している。

iptables -A POSTROUTING -t nat -s $DMZ_LAN -o $EXTERNAL_INTERFACE -j SNAT \
    --to-source $IPADDR
#POSTROUTING チェーンで machine1 から出ていく machine2 の IP アドレスを
#machine1 の IP アドレスに変換

iptables -A PREROUTING -t nat -p tcp --dport 80 -i $EXTERNAL_INTERFACE -j DNAT \
 --to-destination 192.168.150.2
#PREROUTING チェーンで machine1 に入ってくる http パケットの宛先 IP アドレスを
#192.168.150.2 に変換

iptables -A PREROUTING -t nat -p tcp --dport 443 -i $EXTERNAL_INTERFACE -j DNAT \
 --to-destination 192.168.150.2
#PREROUTING チェーンで machine1 に入ってくる https パケットの宛先 IP アドレスを
#192.168.150.2 に変換

################################################################################
##
## 設定の保存
##
#/etc/init.d/iptables save active
```

```
## パケットの転送を開始する
echo 1 > /proc/sys/net/ipv4/ip_forward

exit 0
```

6.2.6 確認用ソフトウェアのインストール

machine1 及び machine2 において以下のコマンドを実行し、ファイアウォール設定の確認時に利用するクライアントソフトウェアをインストールした。

```
yum info lynx
yum install lynx
yum info bind-utils
yum install bind-utils
rpm -ql bind-utils — grep /usr/bin
```

6.2.7 ファイアウォールの動作状況の最終確認

ここで、lynx コマンドや ping コマンドを用いて、machine1 から外部ネットワークへの接続、machine2 から外部ネットワークへの接続、machine3 から外部ネットワークへの接続、machine3 から machine1 への接続、machine3 から machine2 への接続、外部ネットワークから machine1 への接続が可能であることを確認した。

6.2.8 最終状態を保存

以下のコマンドを実行し、最終状態を保存した。

```
iptables-save > /etc/sysconfig/iptables
```

次に machine2 について下記実験を行った。

6.2.9 firewalld の動作状態を確認

以下のコマンドを実行し、firewalld の動作状態を確認した。

```
systemctl status firewalld
```

6.2.10 現在のゾーンの許可状態を確認

以下のコマンドを実行し、現在のゾーンの許可状態を確認した。

```
firewalld-cmd –get-active-zones
```

6.2.11　必要なサービス許可設定の追加

以下のコマンドを実行し、必要なサービス (http や https など) を追加した。

```
firewall-cmd –add-service=dhcpv6-client
firewall-cmd –add-service=http
firewall-cmd –add-service=https
firewall-cmd –list-all
```

6.2.12　動作確認

SSH、HTTP/HTTPS、DHCPv6client、すべての ICMP パケットのみ許可し、その他の接続は全て拒否されることを確認した。

6.2.13　各設定の永続化

以下のコマンドを実行し、各設定の永続化を行った。

```
firewall-cmd –permanent –add-sevice=dhcpv6-client
firewall-cmd –permanent –add-service=http
firewall-cmd –permanent –add-service=https
```

6.2.14　firewall の再起動

以下のコマンドを実行し、firewall の再起動を行った。

```
firewalld-cmd –reload
```

6.3　実験結果

6.3.1　パケット転送の有効化を行った結果

```
[root@icesc16 ~]# sysctl --system
* Applying /usr/lib/sysctl.d/00-system.conf ...
* Applying /usr/lib/sysctl.d/10-default-yama-scope.conf ...
kernel.yama.ptrace_scope = 0
* Applying /etc/sysctl.d/10-ipv4.conf ...
net.ipv4.ip_forward = 1
* Applying /usr/lib/sysctl.d/50-default.conf ...
kernel.sysrq = 16
kernel.core_uses_pid = 1
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.promote_secondaries = 1
net.ipv4.conf.all.promote_secondaries = 1
```

net.ipv4.ip_forward=1 と設定されており、パケット転送が有効化されていることが確認できた。

6.3.2 iptables の設定状況を確認した結果

```
[root@icesc16 ~]# iptables -L -n
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           state NEW udp dpt:67
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           state NEW udp dpt:67
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED

Chain FORWARD (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW,ESTABLISHED tcp dpt:22
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED

Chain OUTPUT (policy DROP)
target     prot opt source               destination
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:80
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:443
ACCEPT     udp  --  0.0.0.0/0            0.0.0.0/0           state NEW udp dpt:53
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     icmp --  0.0.0.0/0            0.0.0.0/0
ACCEPT     tcp  --  0.0.0.0/0            0.0.0.0/0           state NEW tcp dpt:22
ACCEPT     all  --  0.0.0.0/0            0.0.0.0/0           state RELATED,ESTABLISHED
```

各チェーンとその用途を以下の表に示す。

表 3　各チェーンと用途

INPUT	受信したパケットに対するチェーン
OUTPUT	送信するパケットに対するチェーン
FORWARD	転送するパケットに対するチェーン
PREROUTING	自ホストに届いたパケットを一番最初に処理するチェーン
POSTROUTING	自ホストから外部に出て行くパケットを最後に処理するチェーン

policy DROP は iptables にルールが示されていない場合デフォルトで DROP(破棄) 処理するということを表している。target は特定の IP アドレスやプロトコルを条件にパケットに対する処理を表している。(ACCEPT はパケットを受け取ることを表している。)
prot はプロトコルを表しており、opt は処理時のオプションを表している。
source はクライアントの IP アドレス、destination はサーバの IP アドレスを表している。(0.0.0.0/0 は全てのネットワークアドレスを意味する。)
state は通信状態を表しており、NEW は新規の接続を許可し、ESTABLISHED は戻りパケットを許可、

RELATED は前の通信と関連のある接続を許可することをそれぞれ表している。
dpt はプロトコルのポート番号を表している。(22:SSH、53:DNS、80:http、443:https)

6.3.3 iptables のルールを表示した結果

```
root@icesc16 ~]# iptables-save
# Generated by iptables-save v1.4.21 on Thu May 23 14:28:44 2019
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 192.168.200.0/24 -o enp1s0 -j SNAT --to-source 192.168.100.16
COMMIT
# Completed on Thu May 23 14:28:44 2019
# Generated by iptables-save v1.4.21 on Thu May 23 14:28:44 2019
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -i lo -j ACCEPT
-A INPUT -i enp1s0 -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -i enp3s0 -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -i enp3s0 -p udp -m state --state NEW -m udp --dport 67 -j ACCEPT
-A INPUT -i enp3s0 -p icmp -j ACCEPT
-A INPUT -i enp2s0 -p udp -m state --state NEW -m udp --dport 67 -j ACCEPT
-A INPUT -i enp2s0 -p icmp -j ACCEPT
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i enp3s0 -o enp1s0 -p tcp -m state --state NEW,ESTABLISHED -m tcp --dport 22
-j ACCEPT
-A FORWARD -i enp3s0 -o enp1s0 -p icmp -j ACCEPT
-A FORWARD -i enp1s0 -o enp3s0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -o lo -j ACCEPT
-A OUTPUT -o enp1s0 -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A OUTPUT -o enp1s0 -p tcp -m state --state NEW -m tcp --dport 80 -j ACCEPT
-A OUTPUT -o enp1s0 -p tcp -m state --state NEW -m tcp --dport 443 -j ACCEPT
-A OUTPUT -o enp1s0 -p udp -m state --state NEW -m udp --dport 53 -j ACCEPT
-A OUTPUT -o enp1s0 -p icmp -j ACCEPT
-A OUTPUT -o enp3s0 -p icmp -j ACCEPT
-A OUTPUT -o enp2s0 -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
COMMIT
# Completed on Thu May 23 14:28:44 2019
```

大きく nat と filter という機能のルールを表している。nat は IP アドレスの変換、filter はパケットフィルタリングを行っている。nat は PREROUTING、POSTROUTING、OUTPUT、INPUT が存在し、filter は INPUT、OUTPUT、FORWARD というチェーンが存在する。それぞれの用途は表 3 参照。
nat 内のチェーンは全てデフォルトで ACCEPT(パケット受理) になっており、filter 内のチェーンは全てデフォルトで DROP(パケット破棄) になっている。つまり、iptables にルールが書かれてないならば nat では ACCEPT され、filter では DROP される。
-A はルールの追加を表している。
iptables の主なオプションは以下の表に示す。

表 4　iptables の主なオプション

-i	パケットが入ってくるインターフェース名を指定する。
-o	パケットが出ていくインターフェース名を指定する。
-s	クライアント IP アドレスを指定する。
-p	プロトコルを指定する。サービス名またはポート番号で指定可能。
-m state	ステートフルインスペクションで使用する。
-j	ターゲットを指定する。
–dport	ポート番号を指定する。

6.3.4　ファイアウォール動作状況の最終確認結果

- machine1 から外部ネットワークへの接続 (ssh、http/https、dns、ping)
- machine2 から外部ネットワークへの接続 (ssh、http/https、dns、ping)
- machine3 から外部ネットワークへの接続 (ssh、http/https、dns、ping)
- machine3 から machine1 への接続 (ssh、ping)
- machine3 から machine2 への接続 (ssh、ping)
- machine2 から machine3 への接続 (ssh、http/https、ping 拒否)
- 外部ネットワークから machine1 への接続 (ssh)

これらが全て確認できた。

6.3.5　firewalld の動作状態の確認結果

```
[root@localhost ~]# systemctl status firewalld
firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-05-23 13:00:52 JST; 4h 49min ago
     Docs: man:firewalld(1)
 Main PID: 762 (firewalld)
   CGroup: /system.slice/firewalld.service └─
           762 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
```

active となっているため firewall が稼働していることが確認できた。

6.3.6　現在のゾーンの許可状態の確認結果

```
[root@www6 ~]# firewalld-cmd --get-active-zones
public
  interfaces: enp3s0
```

public は公共の場で使用するために定義されたゾーンであることを表しており、enp3s0 インターフェースに割り当てられていることがわかる。

6.3.7　各設定の永続化設定の結果

```
[root@icesc16 ~]# firewall-cmd --permanent --add-service=dhcp6-client
```

```
Warning: ALREADY_ENABLED: dhcpv6-client
success
[root@icesc16 ~]# firewall-cmd --permanent --add-service=http
success
[root@icesc16 ~]# firewall-cmd --permanent --add-service=https
success
```

それぞれの設定を恒久化できたことがわかる。

6.4 考察

firewall ゾーン範囲設定の意味 (?)

7 [課題 5]WWW サービスの設定

7.1 目的と概要

本実験では、Apache WWW サーバ及び https 対応モジュールのインストールと自己証明書の作成により WWW サーバを起動することにより、WWW サーバの起動方法及び通信路暗号化プロトコルである SSL/TLS の設定方法とその重要性を理解する。

7.2 実験方法

machine2 で下記の実験を行った。

7.2.1 firewall の停止

以下のコマンドを実行し、firewall の停止を行った。

```
systemctl stop firewalld
systemctl status firewalld
```

7.2.2 WWW サーバのインストール

以下のコマンドを実行し、Apache WWW サーバと https 対応モジュールをインストールした。

```
yum info httpd
yum info mod_ssl
yum install httpd
yum install mod_ssl
```

7.2.3 SSL/TLS 自己署名証明書の作成

/etc/pki/tls/certs に移動した。
以下のコマンドを実行し、秘密鍵を作成した。

```
openssl genrsa -aes128 1024 > server.key
```

以下のコマンドを実行し、パスフレーズの除去を行った。

```
openssl rsa -utf8 -in server.key -out server.key
```

以下のコマンドを実行し、CSR を作成した。

```
openssl req -utf8 -new -key server.key -out server.csr
```

以下のコマンドを実行し、有効期間 1 年のサーバ証明書を作成した。

```
openssl x509 -in server.csr -out server.crt -days 365 -req -signkey server.key
```

7.2.4 WWW サーバの設定ファイルにサーバ証明書を指定

自己証明書を作成したら、設定ファイル /etc/httpd/conf.d/ssl.conf を以下のように変更した。

```
SSLProtocol +TLSv1.2
SSLCertificateFile /etc/pki/tls/certs/server.crt
SSLCertificateKeyFile /etc/pki/tls/certs/server.key
```

7.2.5 WWW サーバの主設定ファイルを修正

Apache WWW サーバ主設定ファイル/etc/httpd/conf/httpd.conf の ServerName ディレクティブを group6a に変更した。

7.2.6 WWW サーバの起動

以下のコマンドを実行し、WWW サーバを実行した。

```
systemctl restart httpd
```

7.2.7 動作確認

machine3 と外部テスト端末から http、https 両方の動作を確認した。

7.2.8 WWW サーバのブート時自動起動を指定

以下のコマンドを実行し、WWW サーバのブート時自動起動を指定した。

```
systemctl enable httpd
```

7.3　実験結果

7.3.1　machine3 から https サーバに接続した結果

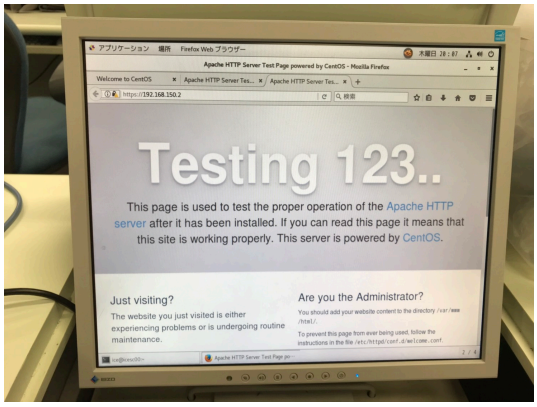

図 1　machine3 から https サーバに接続した結果

7.3.2　外部テスト端末から http サーバ接続した結果

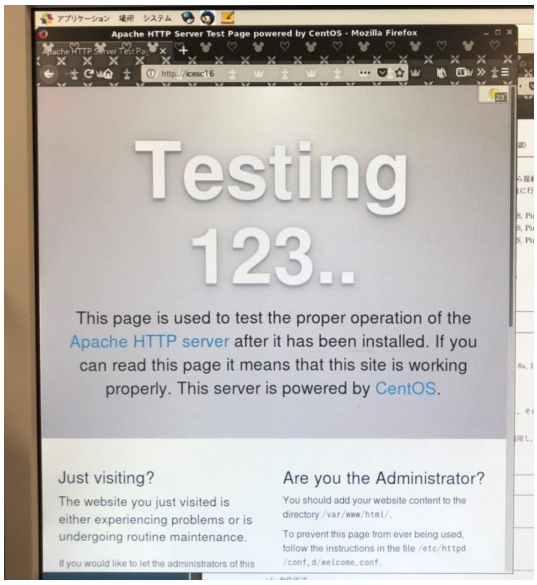

図 2　外部テスト端末から http サーバに接続した結果

図 1、2 のように接続が確認できた。

7.3.3　自己署名証明書の確認

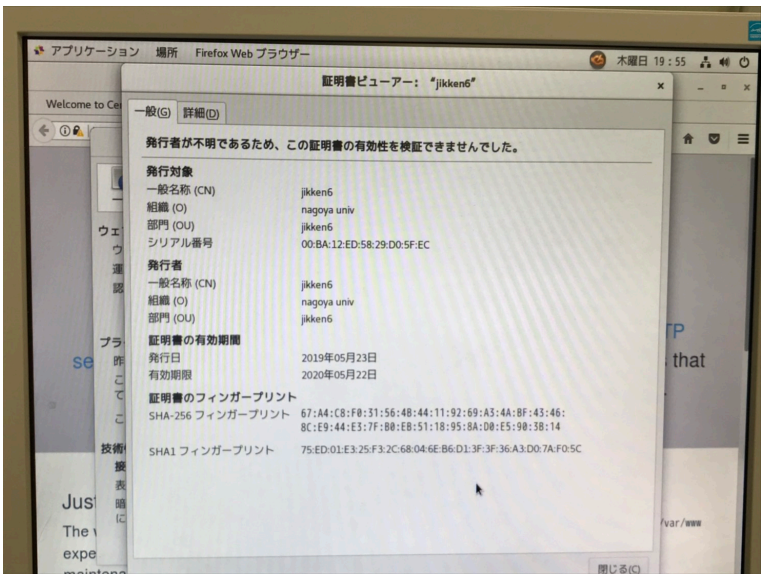

図 3　証明書の詳細

秘密鍵の作成の場面で設定した情報であることが確認できた。

7.3.4　WWW サーバのブート時自動起動を指定した結果

再起動を行い、systemctl status httpd コマンドで確認した結果、active になっていることが確認できた。

7.4　考察

aaaaaaaaaaaaaaaaaaaaa

8　まとめ

本実験では、ローカルネットワークの設定、DHCP サービスの設定、ファイアウォールの設定、WWW サービスの設定について実験的に検証した。実験の結果、各コマンド出力結果の意味、各設定の意味を明らかにした。このネットワークを利用することで WEB アプリケーションを作成することが可能である。今後は、ネットワークに関する理解をさらに深めることで、より高度なネットワーク設定が可能になると思われる。

9　[調査課題 1]TCP パケットのヘッダ情報及び IP パケットのヘッダ情報

ここでは、TCP パケットのヘッダ情報及び IP パケットのヘッダ情報について調査した

9.1　TCP パケットのヘッダ情報

TCP パケットのヘッダ情報の概要は以下の図で示される。

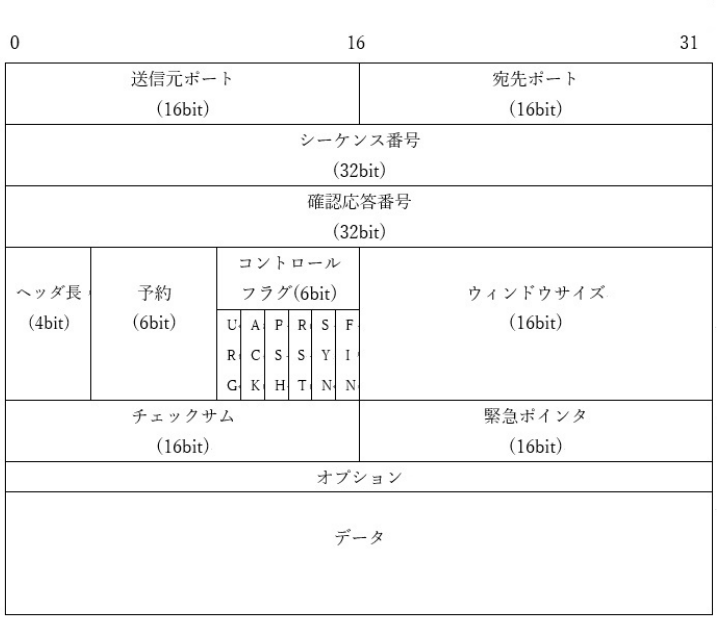

図 4　TCP パケットヘッダ情報の概要

9.1.1　送信元ポート

送信元のアプリケーションを識別するための番号

9.1.2　宛先ポート

宛先のアプリケーションを識別するための番号

9.1.3　シーケンス番号

送信したデータの順序を示す番号で、送信するデータ 1 バイトごとにシーケンス番号を 1 つずつ増やし、2^{32} を超えるとまた同じ値を繰り返す。

9.1.4　確認応答番号

受信したデータに対して受信位置をバイト位置で表すフィールドで、受信が完了したデータ位置のシーケンス番号 +1 を返す。

9.1.5　ヘッダ長

TCP データが始まる位置を表すフィールドで、TCP ヘッダの直後にデータ部が続くため、TCP データの長さを示している。

9.1.6　コントロールフラグ

URG、ACK、PSH、RST、SYN、FIN の 6 つのビットで構成されており、これらのビットは 1 が入るときに意味を成す。

9.1.7　URG

urgent の略で、緊急データが含まれていることを示すフラグ。デフォルトでは 0 が入っており、1 で ON になる。

9.1.8　ACK

acknowledge の略で、有効な ACK 番号が TCP ヘッダに含まれていることを示すフラグ。TCP の 3 ウェイハンドシェイク時の最初を除き、TCP パケットは全て ACK のフラグが ON になっている。

9.1.9　PSH

push の略で、受信したデータをバッファリングせずにすぐにアプリケーションに引き渡すように要求するためのフラグ。

9.1.10　RST

reset の略で、TCP 接続を中断、拒否したい場合にセットされるフラグ。RST フラグを ON にしたパケットを送信することで、現在の TCP 接続を強制終了させることができる。

9.1.11　SYN

synchronize の略で、TCP の 3 ウェイシェイクハンド時のオープン処理の開始に双方のそれぞれが SYN フラグを ON にして ACK 番号を同期させる。以降のパケットにはセットされない。

9.1.12　FIN

finish の略で、TCP 接続を終了させるためセットされるフラグ。双方から FIN フラグが ON である TCP パケットを送信することで TCP 接続は終了する。

9.1.13　ウィンドウサイズ

受信側が一度に受信することができるデータ量を送信側に通知するために使用される。送信側は、この値のデータ量を超えて送信することはできない。

9.1.14　チェックサム

TCP ヘッダとデータ部分のエラーチェックを行うために使用される値が入れられる。

9.1.15　緊急ポインタ

コントロールフラグの URG が 1 である場合にのみ使用されるフィールドであり、緊急データの開始位置を示す情報が入れられる。

9.1.16　オプション

TCP 接続の特性を設定するために利用される可変長のフィールド。(性能を向上させるために利用する)MSS のやり取りなどに用いられる。TCP ヘッダの長さを 32 ビットの整数にするように必要に応じて空データのパディング (0) が埋められる。

9.1.17　データ

TCP のデータ部であり、TCP 接続がタイムアウトしても切断されないようにデータを含まない TCP ヘッダだけのパケットを送る場合もある。

9.2　IP パケットのヘッダ情報

次に、IP パケットのヘッダ情報についてまとめる (文献 [1] 参照)
IP パケットのヘッダ情報の概要は以下の図で示される。

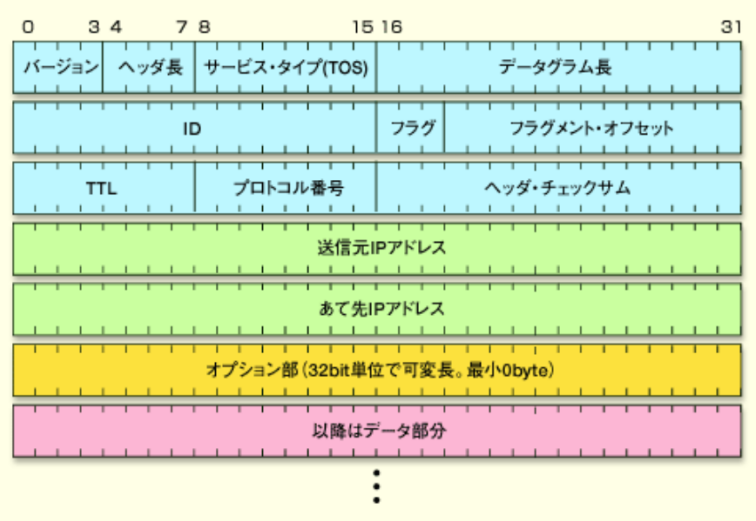

図 5　IP パケットヘッダ情報の概要 (文献 [1] から引用)

9.2.1　バージョン (4bit)

バージョンフィールドは IP プロトコルのバージョンを表現するために使われる。IPv4 を用いる場合はこのフィールドでは常に 4(2 進数では 0010) になっており、IPv6 を用いる場合は常に 6(2 進数では 0110) になっているため、同ネットワーク媒体に IPv4 と IPv6 のパケットは混在していても区別することができる。

9.2.2　ヘッダ長 (4bit)

ヘッダ長フィールドは、IP ヘッダ部分 (固定長部分 + オプション部分) のサイズを表すために使われる。4bit 幅しかないため 0 から 15 までしか表すことができないが、ヘッダの長さは 32bit 単位で数えるため実際の最大長は 60bytes まで表すことが可能である。また、IP ヘッダの固定長部分は常に 20bytes あるためこの

フィールドの最小値は 5 であり、最大値は 15 である。

9.2.3 サービス・タイプ (TOS)(8bit)

サービス・タイプフィールドは IP パケットの優先度などを表す TOS を指定するために使われる。しかし、現在使われている TCP/IP ネットワークではこのフィールドによる TOS 指定はほとんど使用されておらず意味を持っていないことが多い。

9.2.4 データグラム長 (16bit)

データグラム長フィールドは IP パケット全体 (IP ヘッダ部分 + データ部分) のサイズを byte 単位で表すために使われる。このフィールドは 16bit 幅であるため IP パケットのサイズの最大長 (送信可能なデータ + ヘッダ) は 64Kbytes である。このフィールドの最小値は 20+ データ部分の長さ (byte) である。

9.2.5 ID(16bit)

ID フィールドは、IP フラグメンテーションの際に、IP パケットを識別するために使われる。サイズの大きなデータを送信する場合、IP フラグメンテーションによって IP パケットをいくつか分割して小さくしてから送信するが、このフラグメント化されたパケットが後で 1 つの IP パケットに再構成するための目印として使われる。

9.2.6 フラグ (3bit)

フラグフィールドはフラグメンテーションにおいて利用される、特殊なフラグ情報を表すために使われる。3bit 分のデータがあるが実際使われているのは 2bit である。

- MFbit
  MFbit はフラグメントがさらに続くかどうかを表すために使われる。1 つの IP パケットをいくつか分割した場合、最後部のパケットではこの MFbit を 0 にし、その他のパケットでは MFbit を 1 にする。つまりこの bit が 1 ならばフラグメント化された IP パケットがさらに後ろに続くことを表している。
- DFbit
  DFbit は IP パケットを分断してはいけないという指示を与えるために使われる。IP パケットのフラグメント化とその再構成は少なからず複雑な操作が必要になり、性能の低いコンピュータなどではその実現が難しい。こういう場合に DFbit を使うことでフラグメンテーション処理を行わずに TCP/IP 通信を実現することが可能である。

9.2.7 フラグメント・オフセット (16bit)

フラグメント・オフセットフィールドは、フラグメント化された IP パケットにおいてフラグメントのどの部分が IP パケット中に含まれているかを示すオフセット数値を表すために使われる。

9.2.8 TTL(8bit)

TTL フィールドは IP パケットの寿命を表すために使われる。IP パケットを送信するコンピュータはこのフィールドに適当な数値をセットし、その決められた寿命の間だけしか IP パケットが生存できないようにする。

9.2.9　プロトコル番号 (8bit)

プロトコル番号フィールドは、上位のトランスポート層のネットワーク・プロトコルの種類を表す番号を格納するために使われる。

9.2.10　ヘッダ・チェックサム (16bit)

ヘッダ・チェックサムフィールドはヘッダ部分 (固定部分 + オプション部分) のチェックサム (整合性を検査するためのデータ) を表すために使われる。このチェックサム計算では「1 の補数演算」という特別なアルゴリズムが利用される。通常のコンピュータでは「2 の補数演算」によるチェックサムがよく使われるが IP では計算が高速かつ十分な信頼性が確保できる「1 の補数演算」が使われる。

9.2.11　送信元 IP アドレス (32bit)

送信元 IP アドレスフィールドは、送信元のコンピュータの IP アドレスを表すために使われる。

9.2.12　宛先 IP アドレス (32bit)

宛先 IP アドレスフィールドは、IP パケットの送信先コンピュータの IP アドレスを表すために使われる。

9.2.13　オプション (32bit)

オプションフィールドは、IP パケットの送信に伴い、様々な付加的機能を実現するために使われる。通常はオプションは利用されないが、IP パケットの通過のログやルーティングなどで利用される。

10　[調査課題 2]TCP/IP 通信におけるブロードキャストの役割

TCP/IP 通信で必ず指定が必要な IP アドレスはネットワーク部とホスト部からなり、その境界はサブネットマスクによって決められている。ブロードキャストアドレスはリミテッドブロードキャストアドレスとディレクテッドブロードキャストアドレスの 2 種類があり、ディレクテッドブロードキャストアドレスは IP アドレスのホスト部の bit がすべて 1 である IP アドレスであり、リミテッドブロードキャストアドレスは全て 1 の IP アドレスである。ディレクテッドブロードキャストアドレスでは同じネットワーク内にある全てのホストにデータを送信 (ブロードキャスト通信) することができ、リミテッドブロードキャストアドレスでは自身がいるネットワークにある全てのホストにデータを送信することができる。リミテッドブロードキャストアドレスを用いたブロードキャスト通信で送信されたパケットはルータから先のネットワークへ送られることはないが、ディレクテッドブロードキャストアドレスを用いた場合はルータを超えることもできる。リミテッドブロードキャストアドレスはこの性質を用いて、新しく追加した端末が通信先の MAC アドレスを調べるときに使う。通信開始時に LAN 上の全てのコンピュータに対して通信先の MAC アドレスを問い合わせて、対応する相手がそれに答えることで通信相手の MAC アドレスがわかる。しかし、LAN に接続するデバイスの数が増えるとブロードキャスト通信で MAC アドレスを探すのは困難になるため LAN を分割する必要がある。ネットワークアドレスがわかる場合には基本的にディレクテッドブロードキャストアドレスを用いることが多い。(文献 [4]、[5] 参照)

11 [調査課題 3]TCP パケット送信の過程

まず接続が開始の合図がされるとデータに調査課題 1 で示したような TCP ヘッダを付随する。ここで TCP ヘッダには SSH プロトコルに従うことが定められている。この段階をトランスポート層という。TCP ヘッダがつけられると、次にここに調査課題 1 で示したような IP ヘッダが付随する。ここで宛先の IP アドレス及び送信元の IP アドレスが指定されている。この段階をネットワーク層という。IP ヘッダがつけられると、ここにイーサネットヘッダが付随する。ここで送信元 MAC アドレスおよび宛先 MAC アドレスが指定されている。この宛先 MAC アドレスは最寄りのルータ (ここでは machine1) の MAC アドレスである (ここでは 00:e0:67:12:2d:b4)。さらにここでこれらのデータを電気信号に変換する。この段階をネットワークインターフェース層という。最寄りのルータに届けられたこれらの情報をもとに次に転送すべきルータを探す。目的のルータに直接つながっていればそこに送られるが、繋がっていない場合は、ルーティングテーブルを元に転送すべきパケットのヘッダを次のルータ (ここでは FW 構築実験用ルータ) の MAC アドレスに書き換える。ここで、ルーティングテーブルには MAC アドレスが書かれていないためブロードキャスト通信で対象のルータを探し、MAC アドレスを調べる。そして FW 構築実験用ルータから IP ヘッダの宛先 IP アドレスを元に PC2 にデータが届けられる。(文献 [12] 参照)

12 [調査課題 4] 各サービスにおけるセキュリティ強化の動向

12.1 Unix/Linux システムのユーザ認証

マルチユーザーシステムである Unix/Linux は複数のユーザーが同時に 1 台のコンピュータを利用することができ、その複数のユーザを管理することもできる。Linux/Unix システム認証では、ユーザが SGD ホスト上に Linux/Unix システムアカウントを保持していれば SGD にログインできるようにする。ユーザーを Linux/Unix システムのユーザーデータベースと照合して認証し、ユーザー識別情報プロファイルを決定する。SGD のログイン画面で、ユーザーはユーザー名とパスワードを入力する。SGD はローカルリポジトリ内でユーザーが入力した内容に一致する名前属性を持つユーザープロファイルを検索する。一致する人物オブジェクトがない場合ログイン名属性を対象に、最後に電子メールアドレス属性を対象に検索を繰り返す。ユーザープロファイルが見つからない場合、次のログイン認証メカニズムが試される。ユーザープロファイルが見つかると、そのオブジェクトのログイン名属性が Unix/Linux システムユーザー名として使用される。このユーザー名およびユーザーの入力したパスワードが Unix/Linux システムユーザーデータベースと照合される。認証が失敗した場合は、次の認証メカニズムが試される。認証が成功してもユーザープロファイルのログイン属性が有効になっていない場合、ユーザーはログインできず、他の認証メカニズムが試されることはない。認証は成功して、ユーザープロファイルのログイン属性が有効になっている場合にユーザーはログインできる。また、SGD は PAM をサポートしている。Unix/Linux システム認証では、ユーザー認証、アカウント操作、およびパスワード操作に PAM が使用される。PAM はあるサービスに対するユーザー認証を行うための API である。PAM のメカニズムでは下位レベルにある様々な認証スキームを上位レベルの 1 つの API に統合しているため認証を必要とするプログラムを構成する場合ベースにある認証スキームを意識する必要がない。PAM の大きな特徴は認証を動的に構成できることである。PAM を構成することによって、特定のプログラムによるユーザー認証権限を拒否したり、特定のプログラムが認証を行おうとすると警告したりすること

ができる。PAM プログラムは PAM モジュールを使用する。つまり、PAM モジュールが機能するためにアプリケーションの実行時には PAM モジュールがアプリケーションに組み込まれている必要がある。(文献 [8] 参照)

12.2 DNS

DNS サーバーには主に権威 DNS サーバーとキャッシュ DNS サーバーの 2 種類がある。権威 DNS サーバーは DNS コンテンツサーバーとも呼ばれ、ドメイン名に関する完全な情報を格納している。対象となるドメイン名への問い合わせに回答する役割がある。ドメイン名の登録者や、その運用を預かるレンタルサーバー事業者などが運用している。キャッシュ DNS サーバーは権威 DNS サーバーに名前解決の問い合わせをする。次に同じ問い合わせを受けた時に備え、データを一時保存するという機能を持っている。キャッシュ DNS サーバーは ISP(プロバイダ) などが会員へ提供するために運用していたり、社内ユーザ用に企業内で運用されていたりするが、十分な管理・運用がされていないことが多く、攻撃者に狙われやすい傾向がある。中間者攻撃による DNS データの盗聴や改ざんを防ぎ、ユーザーのプライバシーとセキュリティを向上させるために DoT(DNS over TLS) がある。これは TLS を介した DNS の回答の暗号化とラッピングのためのセキュリティプロトコルである。また、DNS はキャッシュポイズニングも問題になっている。これは、偽の再帰的な問い合わせに対して、本物のコンテンツサーバの回答よりも先に偽の回答を送り込むことで、キャッシュサーバに偽の情報を覚えこませる攻撃である。これにより、クライアントは提供された偽の情報によって攻撃者が罠を張った web サーバに誘導されてしまう危険性や、データサイズの大きな偽の複製を覚えこませた後、キャッシュサーバに対して発信元を詐称した再帰的な問い合わせを行うことでデータサイズの大きな偽の複製を攻撃対象サーバに送信してしまう危険性がある。これを防ぐために whois サービスに登録されている DNS サーバと nslookup コマンドで表示される DNS サーバが一致しているか確認する方法がある。whois サービスとは IP アドレスやドメイン名の登録者などに関する情報をインターネットユーザがだれでも参照できるサービスである。このサービスを使用することでドメインの情報管理を行う DNS サーバを確認できる。(文献 [9]、[10] 参照)

12.3 メール配信サービス

メール配信サービスにおけるセキュリティ強化の方法として SPF や DKIM がある。SPF はメールの送信元を詐称するなりすましメール対策の一つである。差出人のメールアドレスが詐称されていないかどうかを確認できる。携帯向けの配信を行う場合、SPF を設定していないサーバーからメールを送ると、受信が拒否される可能性が高くなる。メールの送信元になるサーバーの IP アドレスを DNS に登録し、その送信元サーバーからメールが送られると、メールを受信したサーバーは送信元が設定しているアドレスと一致するかどうか判別を行う。一致していない場合は別のサーバーからなりすまされて送信されたメールであると判別して、拒否される。DKIM とは迷惑メール対策となる送信ドメイン認証技術である。DKIM はメール送信者のドメインを証明する作成者署名をつけることができるため、この作成者署名をメールに付与することでより信頼性の高いメールを送信できる。(文献 [11] 参照)

13　[調査課題 5]SSL/TLS を用いるプロトコル

まず、SSL とは「インターネット上でデータを暗号化して送受信する方法のひとつ」(文献 [6] より引用) であり、TLS とは「SSL をもとに標準化させたもの」(文献 [6] より引用) である。通常、インターネットでは暗号化されずにデータが送信されているが、通信途中でデータを傍受されると情報が第三者に漏れてしまう可能性がある。また、相手のなりすましに気づかず通信すると、なりすましの相手にデータを送信してしまうことになる。現在クレジットカード番号や個人情報を多く扱う web サイトでは通信途中の傍受やなりすましによる情報漏洩を防ぐために SSL/TLS によってデータを暗号化して送受信している。SSL/TLS に対応した web ブラウザを利用して SSL/TLS で保護されたサイトに接続すると、そのサイトの web サーバーはサーバー証明書を web ブラウザに送る。web ブラウザはこのサーバー証明書に付された認証局の署名が正しいものであることを、事前に安全に入手していた認証局のルート証明書を用いて検証する。検証ができたら、web ブラウザは web サーバーと暗号化通信を行うための鍵を交換し、その鍵を用いて通信データが自動的に暗号化されるようになる。

メールの送受信用の POP3、IMAP4、SMTP プロトコルではメール本文や認証用パスワードが暗号化されないため、盗聴によって悪用される恐れがある。この欠点を SSL で解消したプロトコル POP3s、IMAP4s、SMTPs について調査してまとめる。SSL は web 専用の技術でなく、TCP/IP を利用するアプリケーション全般で利用が可能である。POP3s/IMAP4s/SMTPs ではメーラとメール・サーバ間を行き交うデータが SSL によって暗号化される。メール本文などのデータはもちろん認証用のパスワードも安藤化の対象となる。もし第三者が POP3s/IMAP4s/SMTPs の通信を傍受しても暗号が破られない限り、通信パケットからメール本文やパスワードを読み取ることはできない。また POP3s/IMAP4s/SMTPs は、偽のメール・サーバに接続するなりすましを防ぐこともできる。POP3s/IMAP4s/SMTPs にはその FQDN(完全修飾ドメイン名) を証明するデジタル証明書が備わっている。もし、ウイルスやファーミングなどによってメーラにおけるメール・サーバの FQDN の名前解決が偽装され、不正なメール・サーバに向けられた場合でも POP3s/IMAP4s/SMTPs では SSL によって接続時にサーバ側のデジタル証明書が確認されるため、偽のメール・サーバとの接続確立を防ぐことが可能である。同様に、メール・サーバとの送受信内容の改ざんを防ぐ効果もある。POP3s/IMAP4s/SMTPs の実装方式は大きく 2 種類あり、1 つは最初から SSL によって POP3s/IMAP4s/SMTPs を付随した状態で通信を始める方式で、もう一つは最初に POP3/IMAP4/SMTP で通信を始めて途中で SSL に切り替えるという方式である。前者の方式が over SSL で後者の方式が STLS/STARTTLS と呼ばれている。これらの方式の違いでエンド・ユーザに影響があるのは通信時の TCP ポート番号の違いである。前者の場合メーラにてポート番号を POP3s/IMAP4s/SMTPs に割り当てなければいけない。後者の場合は、POP3/IMAP4/SMTP4 と同じポート番号のままで済む。(下表 5 参照)

表 5　プロトコルとポート番号

暗号化なし		暗号化あり	
プロトコル名	ポート番号	プロトコル名	ポート番号
POP3	110	POP3s	995
IMAP4	143	IMAP4s	993
SMTP	25/587	SMTPs	465

POP3s ／ IMAP4s ／ SMTPs によって暗号化されるのは、自分の PC メーラ及び直接接続するメール・サーバ間の通信に限られる。メール・サーバ同士の配送は一般的に暗号化されず相手のメーラとメール・サーバ間の通信が暗号化されるか否かは相手のメーラ/サーバなどに依存し、自分側で制御する術はない。これはメールそのものではなく通信路を暗号化しているデメリットともいえる。自分のメーラから相手のメーラまで一貫して暗号化したい場合はメールそのものを暗号化するなど別の手段を検討すべきである。POP3s ／ IMAP4s ／ SMTPs を利用するには、メーラとメール・サーバの両方ともこれらのプロトコルに対応している必要がある。メール・サーバ側では通常、従来の POP3 ／ IMAP4 ／ SMTP を利用可能にしたまま POP3s ／ IMAP4s ／ SMTPs の対応が追加される。そのため POP3 ／ IMAP4 ／ SMTP でメールを送受信している限り、POP3s ／ IMAP4s ／ SMTPs の対応に気づくことはない。メール・サーバに比べてメーラの方が POP3s ／ IMAP4s ／ SMTPs 対応は進んでおり (Yahoo!メール、Gmail など)、すでに非対応の製品のほうが少ない。(文献 [7] 参照)

参考文献

[1] atmarkIT IP パケットの構造と IP フラグメンテーション
https://www.atmarkit.co.jp/ait/articles/0304/04/news001_2.html(2019 年 5 月 22 日参照)

[2] 日経 xTECH マウント
https://tech.nikkeibp.co.jp/it/article/Keyword/20070207/261214/(2019 年 5 月 25 日参照)

[3] IBM
https://www.ibm.com/developerworks/jp/linux/library/l-lvm2/index.html(2019 年 5 月 25 日参照)

[4] 日経 xTECH ネットの本質をつかむ 10 大ポイント
https://tech.nikkeibp.co.jp/it/members/NNW/ITARTICLE/20010406/1/(2019 年 5 月 26 日参照)

[5] 日経 xTECH 2 種類のブロードキャストとサブネット・マスクを理解しよう
https://tech.nikkeibp.co.jp/it/article/COLUMN/20070808/279348/(2019 年 5 月 26 日参照)

[6] 総務省 SSL/TLS の仕組み
http://www.soumu.go.jp/main_sosiki/joho_tsusin/security_previous/kiso/k01_ssl.htm(2019 年 5 月 29 日参照)

[7] atmarkIT メールの送受信を暗号化する POP3s ／ IMAP4s ／ SMTPs とは
https://www.atmarkit.co.jp/ait/articles/0801/18/news126.html(2019 年 5 月 29 日参照)

[8] IBM Linux
https://www.ibm.com/developerworks/jp/linux/library/l-pam/index.html(2019 年 5 月 29 日参照)

[9] マイナビ 第 1 回〜深刻化する DNS への攻撃〜
https://news.mynavi.jp/article/dns_security-2/（2019 年 5 月 29 日参照)

[10] IPA DNS キャッシュポイズニング対策
https://www.ipa.go.jp/files/000017289.pdf(2019 年 5 月 29 日参照)

[11] Cuenote
https://www.cuenote.jp/fc/(2019 年 5 月 29 日参照)

[12] IT 人材ラボ ルーティングの動作とルーティングテーブル
https://itjinzai-lab.jp/article/detail/217(2019 年 5 月 29 日参照)