サイバーセキュリティ演習のためのシステム構築 [配布用]

注：　本稿では演習システムのドメイン名を「ysato.net」としたが、このドメイン名は任意である。演習システムを構築する際は、本稿に示す「ysato.net」を「hoge.com」や「foo.xyz」などのように各自で適当なドメイン名に置換することが望ましい。なぜならば、他人とドメイン名が重複すると、図 A のように複数の演習システムを結合したときに正しく名前解決が出来ないためである。

また、ルータのインターネット側ポート Fa0 の IP アドレス「192.168.0.2」も「192.168.0.123」などのように、他の演習システムの IP アドレスと重複しないように設定する。

さらに、複数の演習システムを結合した状態で動作させるためにはルータのポートフォワーディングを設定する必要がある。ただし、本稿にはポートフォワーディングの設定手順が含まれていない (忘れていました) ため、自習して設定されたい (ごめんなさい)。

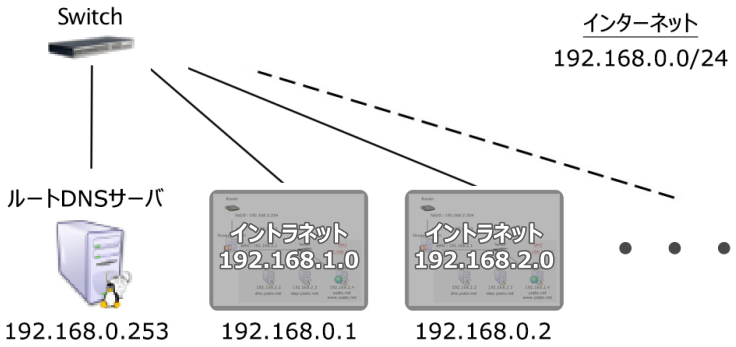

図 A　複数の演習システムを結合したときの構成図

目次

1　はじめに

身の回りのあらゆるものがネットワークに繋がるようになり、人とネットワークの繋がりはますます密接なものになりつつある。例えば、洗濯機や冷蔵庫がネットワークに接続されるスマート家電は、スマートフォンによる戸外からの遠隔操作が可能である。また、2020 年の東京オリンピック・パラリンピックを控え、日本を標的としたサイバー攻撃の増加が懸念されている。従って、現在、サイバーセキュリティ対策は極めて重要な課題である。

そのため、本稿では、サイバーセキュリティに関する実践的な演習が可能なシステムを設計・構築する。構築するシステムは、イントラネットの構成をモデルにすることで、実際のセキュリティインシデント対応に似た演習を行うことができる。また、サーバマシンには Raspberry Pi 2 を使用することで、安価かつ省スペースなシステムを実現することができる。

2　イントラネット

イントラネット (Intranet) は、企業内ネットワークを意味する用語である。イントラネットでは、インターネットと同じ技術を用いて、企業の利便性を高めるためのネットワークが構築される。図 1 に、企業のネットワーク構成例 [1] を示す。

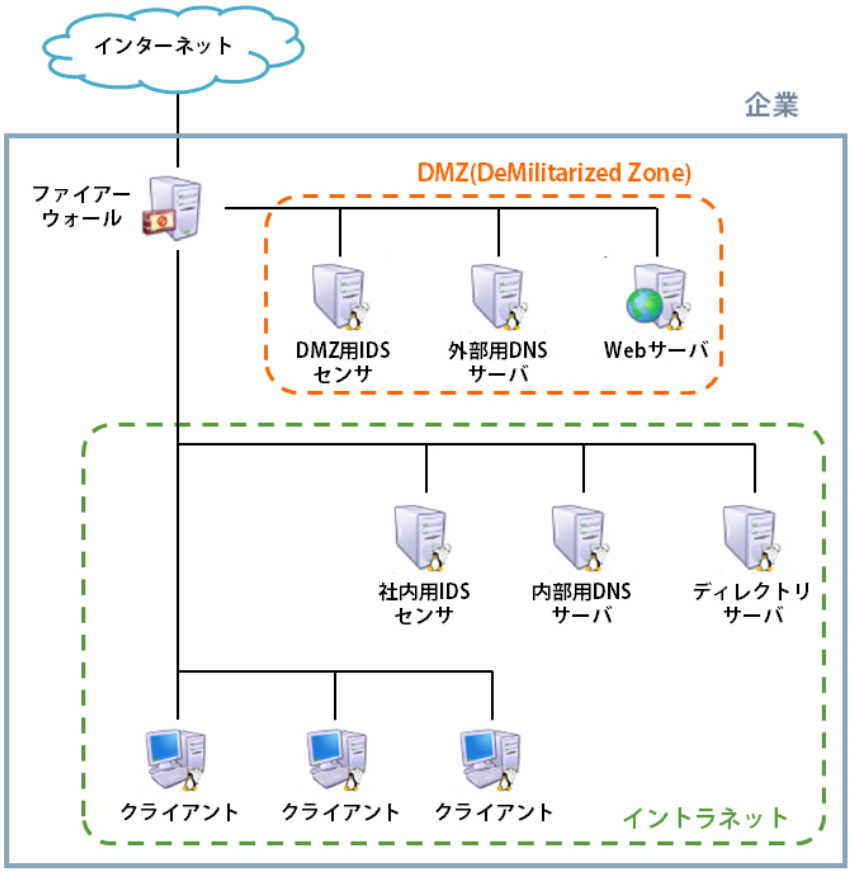

図 1　企業のネットワーク構成例

以下に、イントラネットで用いられる技術についての解説を示す。

2.1　サーバの公開

インターネット上で企業の情報を公開するために、企業が自社のホームページを開設することがある。これには、Web サーバや DNS サーバの設置が不可欠である。また、企業の内外で IP アドレスを変換する仕組みも必要となる。

2.1.1 Web サーバ

Web サーバは、クライアントのコンピュータに対して、HTML ドキュメントなどの Web コンテンツを提供するサーバである。この Web コンテンツの送受信には、HTTP という通信プロトコルが標準的に用いられる。

また、Web サーバ上でプログラムを実行し、動的に Web コンテンツを生成することが可能である。

2.1.2 DNS サーバ

DNS(Domain Name System) サーバは、ドメイン名と IP アドレスの対応関係を管理するサーバである。DNS サーバは、コンテンツサーバとキャッシュサーバの 2 種類に大別される。

コンテンツサーバは、ゾーンと呼ばれるドメイン名と IP アドレスの対応表を保持し、このゾーン情報に基づいて、問い合わせに対する名前解決 (ドメイン名と IP アドレスの変換) を行う。ゾーンには正引きゾーンと逆引きゾーンがあり、ドメイン名を IP アドレスに変換するためのゾーンを正引きゾーンという。逆に、IP アドレスをドメイン名に変換するためのゾーンを逆引きゾーンという。

これらのゾーンは、レコードと呼ばれるデータの集合によって定義されている。表 1 に主要なレコードの種別と意味を示す。表中のホスト名とは、ゾーン内でコンピュータを識別するための名前である。

表 1　主要なレコードの種別と意味

種別	意味
SOA	ゾーン情報の定義
NS	DNS サーバ名
A	ホスト名に対する IP アドレス
PTR	IP アドレスに対するホスト名
CNAME	ホスト名の別名

コンテンツサーバが自身の保持する情報を元に名前解決を行うのに対して、自身では情報を保持しない種類の DNS サーバをキャッシュサーバと呼ぶ。問い合わせを受けたキャッシュサーバは、名前解決が完了するまで反復的にコンテンツサーバへの問い合わせを行い、最終的な結果のみを回答する。同時に、キャッシュサーバは回答を一定期間キャッシュすることで、名前解決のための通信を最小限に抑えることができる。

このように、DNS サーバはコンテンツサーバとキャッシュサーバに分類される。しかし、DNS サーバにビューを定義することで、問い合わせ元に応じて DNS サーバの種類を変えることが可能となる。そのため、1 つのサーバ内でコンテンツサーバとしてもキャッシュサーバとしても振る舞うことができるようになる。

また、DNS サーバのフォワーダ機能では、自身で解決できない要求をフォワーダに指定した DNS サーバに転送することが可能である。

2.1.3 NAT・IP マスカレード

NAT(Network Address Translation) は、2 つのネットワーク間で通信するために、IP アドレスの自動変換を行う技術である。また、IP アドレスに加えて、ポート番号の変換を行う技術を IP マスカレード (IP masquerade) という。

これらの技術は、グローバル IP アドレスを持つインターネット上のクライアントが、プライベート IP アドレスを持つイントラネット内のサーバと通信を行うときなどに用いられる。

2.2 セキュリティ対策

イントラネットは、インターネットと相互の技術を用いて構成されるため、外部からの侵入に対して脆弱である。そのため、ファイアーウォールや DMZ などを用いたセキュリティ対策が重要となる。

2.2.1 ファイアーウォール

ファイアーウォールは、設定されたポリシーに従って、ネットワークからの通信を許可または遮断するシステムである。ファイアーウォールは、OSI 参照モデルのネットワーク層で動作するパケットフィルタリング型やアプリケーション層で動作するアプリケーションゲートウェイ型などに分けられる。パケットフィルタリング型では、通信の IP アドレスやポート番号によって通信を遮断するか許可するかの判断を行う。対して、アプリケーションゲートウェイ型では、通信のデータ部分を参照するため、より細かい通信制御が可能となる。

これらのファイアーウォールを導入することで、外部からの不要な通信を遮断するなどのセキュリティ対策が可能になる。

2.2.2 DMZ

インターネットに公開するサーバの配置を目的とした、インターネットともイントラネットとも隔離されたネットワークを DMZ(DeMilitarized Zone) と呼ぶ。

図 2 に示すように、DMZ はファイアーウォールによってネットワーク的に隔離される。そして、インターネットから直接通信可能な範囲を DMZ に限定することで、イントラネットには外部からの攻撃が及ばないようにする。また、インターネットと DMZ の双方に対してイントラネットからの通信を許可し、イントラネットに属するクライアントの利便性を損なわないようにする。従って、DMZ を設けることで、安全性と利便性の両立が実現される。

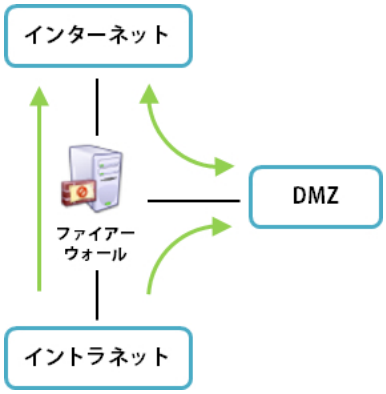

図 2　ファイアーウォールによる通信制御

2.2.3 IDS・IPS

IDS(Intrusion Detection System) は、侵入検知システムである。IDS には、ネットワークを流れる通信を監視するネットワーク型 IDS(NIDS) と、ホストへの侵入やファイル改ざんを検知するホスト型 IDS(HIDS) がある。

侵入の検知のみを行う IDS に対して、通信の遮断を可能にしたシステムが IPS(Intrusion Prevention System) である。IPS では、攻撃と判断された通信をリアルタイムに遮断することで、攻撃を未然に防ぐことが可能となる。

一般的に、IDS や IPS は、ファイアーウォールと併用され、ファイアーウォールで防ぐことができなかった攻撃を検知・遮断するために用いられる。

2.2.4 ディレクトリサーバ

ディレクトリサーバは、ネットワーク上のリソースを管理するサーバである。ディレクトリサーバを使用することで、ネットワーク上に分散した機器の資源管理やユーザアカウントの認証管理を一元化することが可能となる。つまり、管理者によるリソース管理が簡単になると同時に、ユーザは一度の認証によって複数のリソースが利用可能になる。

また、リソースに対する権限の管理が可能で、イントラネット内部からの不正アクセス対策としても有効に活用することができる。

3 脆弱性攻撃

ソフトウェアにおける脆弱性は、プログラムの実装ミスや設計上の欠陥によって発生する。攻撃者によってこれらの脆弱性が悪用されると、システムのマルウェア感染や情報漏洩などの被害が生ずる恐れがある。本節では、Web アプリケーションにおける脆弱性を悪用した代表的な攻撃手段について解説する。

3.1 SQL インジェクション

SQL インジェクションは、プログラムからデータベースを操作する場合に起こり得る脆弱性を利用した攻撃である。特に Web アプリケーションにおいては、Web アプリケーション利用者が指定したキーワードを動的に SQL 文に組み込んでデータベースを操作することがある。

例えば、利用者が入力した文字列を元に検索を行うとき、次のような SQL がデータベースに発行される。

```
SELECT * FROM user WHERE name = '検索文字列';
```

この SQL 文は、name カラムの値が検索文字列と一致するレコードを取得する。ここで、検索文字列として「' OR '1' = '1」という文字列が与えられたとすると、SQL 文は次のようになる。

```
SELECT * FROM user WHERE name = '' OR '1' = '1';
```

この SQL 文は、name カラムの値が空、もしくは'1' = '1' となるレコードを取得する。すなわち、この SQL 文の条件は常に真となるため、user テーブルに格納されている全てのレコードが取得される。

以上が SQL インジェクションと呼ばれる攻撃の原理である。この攻撃においては、入力を適切にエスケープ処理することが基本的な対策になる。

3.2 XSS

XSS(Cross Site Scripting) は、動的に作成される HTML に任意のスクリプトを埋め込むことができる脆弱性を利用した攻撃である。この脆弱性は、Web アプリケーション利用者の入力値をそのまま HTML に組み込んで配信するプログラムに存在する。

例として、次の HTML コードを考える。

```
<p> 入力値 の結果を表示します。 </p>
```

このコードにおいて、入力値には Web アプリケーション利用者が入力する任意の値が入る。ここで、入力に「<script>alert(”XSS”)</script>」という JavaScript コードが与えられたとすると、HTML コードは次のようになる。

```
<p> <script>alert("XSS")</script> の結果を表示します。 </p>
```

この HTML コードが含まれる Web ページをブラウザで表示すると、JavaScript によってアラートダイアログが表示される。

このように、XSS においては、利用者のブラウザ上で任意のスクリプトが実行される可能性がある。これを防ぐためには、入力を適切にエスケープ処理してから HTML に組み込むといった対策が有効である。

3.3 CSRF

CSRF(Cross Site Request Forgeries) は、Web アプリケーション利用者の意図しない操作を、別の Web アプリケーションに対して実行させる攻撃である。

例えば、Web アプリケーション上のスクリプトを実行可能な URL を含むメールが、攻撃者によって送信されたとする。このメールの受信者が、メールに含まれる URL のリンクをクリックすると、受信者は意図せず Web アプリケーション上のスクリプトを実行することになる。

このスクリプトの内容によっては、掲示板に書き込みをさせられたり、オンラインショップでの買い物をさせられたりする可能性がある。CSRF の対策としては、外部サイトからのリクエストを拒否するようにすることなどが挙げられる。

4　システムの設計

本節では、サイバーセキュリティに関する実践的な演習が可能なシステムを設計する。

4.1　ネットワーク構成

演習システムのネットワーク構成を図 3 に示す。

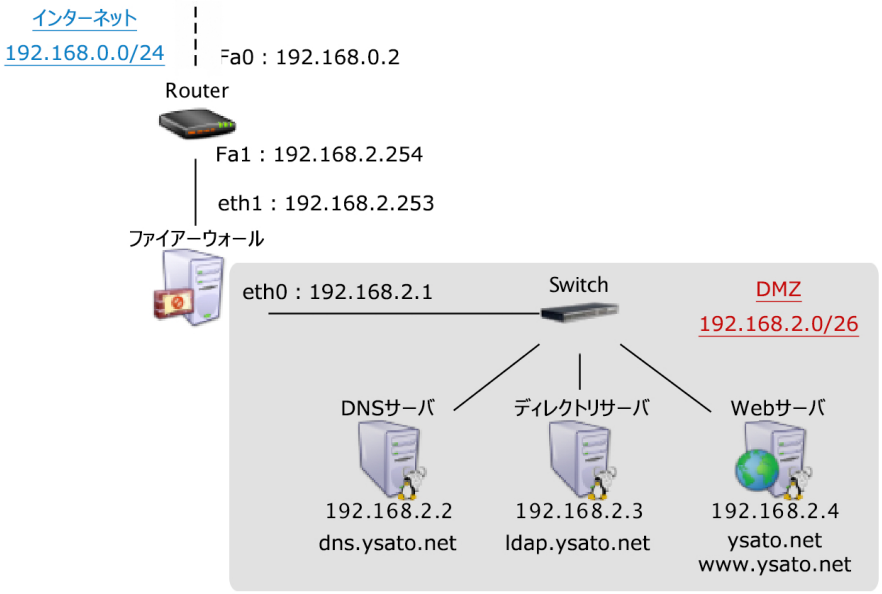

図 3　演習システムのネットワーク構成

演習システムでは、ネットワークを 4 つのサブネットに分けて運用する。表 2 にサブネットの構成を示す。

表 2　演習システムのサブネット構成

ネットワークアドレス	役割
192.168.2.0/26	DMZ
192.168.2.64/26, 192.168.2.128/26	イントラネット
192.168.2.192/26	ファイアーウォール - ルータ間

ネットワークアドレス 192.168.2.0/26 は DMZ に割り当てる。192.168.2.64/26 と 192.168.2.128/26 はイントラネットに割り当てる。192.168.2.192/26 はファイアーウォールとルータ間に割り当てるネットワークアドレスである。

4.2 システム環境

演習システムを構成する主要な機器を表 3 に示す。

表 3　演習システムの使用機器

役割	製品名	製造元
サーバマシン	Raspberry Pi 2 Model B	Raspberry Pi Foundation
サーバのストレージ	MicroSDHC 32GB	Team
ルータ	Cisco1812J	Cisco
スイッチ	EHC-G05PA-B	ELECOM
USB-LAN アダプタ	USB-LAN100R	PLANEX

4.3 マスターイメージ

本稿では、各サーバの構築作業にかかる負担を減らすために、全てのサーバに共通する設定があらかじめ行ってあるマスターイメージを作成する。以下に、マスターイメージの構成を示す。

動作する OS

CentOS7.2.1511.el7 を使用する。

ユーザ名とパスワード

ユーザ名は「root」、パスワードは「centos」とする。

導入するソフトウェア

導入するソフトウェアを表 4 に示す。

表 4　導入するソフトウェア

役割	ソフトウェア名・パッケージ名
Web サーバ	httpd, mariadb, mariadb-server, php-mysql
DNS	bind, bind-utils
ディレクトリサービス	openldap-servers, openldap-clients, php-ldap
IDS・IPS	Snort, DAQ, libpcap, pcre, libdnet, libnfnetlink, libmnl, libnetfilter_queue
エディタ	emacs, vim
ユーティリティ	wget, ntp
開発ツール	Development Tools, make, zlib
プログラミング言語	perl, php, ruby, java-1.8.0-openjdk

ソフトウェアのインストールには基本的に yum を使用する。ただし、Raspberry Pi 2 Model B のアーキテクチャ (ARMv7) に対応するパッケージが存在しない場合は、手動にてコンパイルとインストールを行う。また、開発者向けのソフトウェアがまとめられた Development Tools パッケージグループをインストールすることで、コンパイルに必要な環境を構築する。

4.4 DNS サーバ

DNS サーバは、広く普及している BIND9 を使用し、コンテンツサーバとして構築する。また、IPv6 を除くネットワークからの利用を許容し、全てのネットワークからの名前解決要求に応えるビューを external ビューとして定義する。external ビューの定義を表 5 に示す。

表 5　external ビューの定義

項目	設定
ビュー名	external
ビューが適用される範囲	全てのネットワーク
ゾーン転送	無効
再起問い合わせ	無効
正引きゾーン	ysato.net
逆引きゾーン	2.168.192.in-addr.arpa
正引きゾーンファイル名	ysato.net.wan
逆引きゾーンファイル名	2.168.192.in-addr.arpa

表 6 に、DNS サーバが解決するドメイン名と IP アドレスの対応を示す。

表 6　ドメイン名と IP アドレスの対応

機能	ドメイン名	IP アドレス
DNS サーバ	dns.ysato.net	192.168.2.2
ディレクトリサーバ	ldap.ysato.net	192.168.2.3
Web サーバ	ysato.net www.ysato.net (CNAME)	192.168.2.4

4.5　ディレクトリサーバ

ディレクトリサーバは、オープンソースソフトウェアの OpenLDAP を使用し、架空の会員情報の管理・提供を行う。

サイバーセキュリティ演習システムには、システムの共同開発者である***が設計と実装を担当したディレクトリサーバを使用する。そのため、本稿では、ディレクトリサーバに対する具体的な設計や実装方法は示さず、サーバ間の連携に最小限必要な事項についてのみ示す。ディレクトリサーバに関する詳細については、***の報告書を参照されたい。

4.6　Web サーバ

Web サーバでは、Web アプリケーションの脆弱性を体験できるサイト (脆弱サイト) を提供する。脆弱サイトは、会員制のサイトで、会員データベースの検索機能を持つように構築する。また、脆弱サイト上では、SQL インジェクションや XSS などの代表的な脆弱性攻撃を実行できるようにする。

脆弱サイトの実装には、サーバサイドスクリプト言語の PHP とデータベースシステムの MariaDB を主に使用する。また、脆弱サイトへのログインにはディレクトリサーバの LDAP を使用し、ログイン状態の管理には Cookie を使用する。さらに、Web サーバソフトウェアの Apache2 を使用し、外部のブラウザから `http://192.168.2.4/exploit_site/` でアクセスできるようにする。

図 4 に脆弱サイトの構成を示す。

まず、脆弱サイト利用者は index.php にアクセスする。そして、会員名とパスワードを入力し、ログインを行う。ログイン時には、ldapserver.php に定義された関数を呼び出して認証を試行する。

認証に成功すると、会員検索フォームを提供する main.php に遷移する。main.php で会員検索が実行されると、database.php がデータベースから検索条件に一致する会員を取得し、検索結果が search.php に表示される。

また、logout.php にアクセスすることでログアウトが完了する。

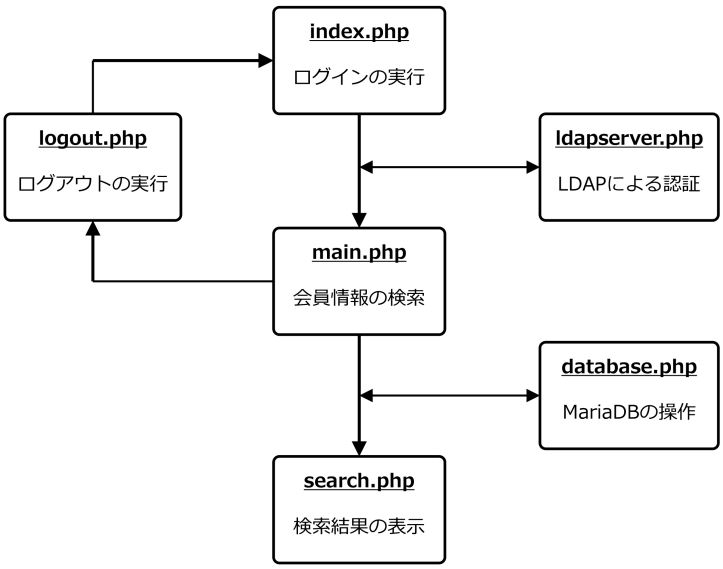

図 4　脆弱サイトの構成

脆弱サイトで使用するデータベース名を web とする。web データベースには、ID、名前、電話番号、パスワードを格納するための user テーブルを定義する。表 7 に user テーブルの定義とレコードを示す。

表 7　user テーブルの定義とレコード

id	name	tel	pass
1	hogeo	1001	hoge
2	pugeo	1012	puge
3	taro	1008	taro
4	fooko	1003	foo
5	barko	1005	bar

ただし、脆弱サイトのログイン認証は LDAP に委譲するため、このデータベースが実際のログイン処理に使用されることはない。これは SQL インジェクションを実行するために使用されるデータベースである。

4.7　ファイアーウォール

ファイアーウォールは、CentOS7 標準のファイアーウォールサービスである firewalld を使用し、パケットフィルタ型として構築する。しかし、Raspberry Pi 2 Model B の LAN ポートは 1 つしかないため、USB-LAN アダプタを使用した LAN ポートの拡張を行う。Raspberry Pi 2 Model B 標準の NIC を eth0、USB によって拡張された NIC を eth1 として、表 8 に示すようにゾーンを設定する。

表 8　firewalld のゾーン定義

ゾーン名	インタフェース	許可するサービス	IP マスカレード
dmz	eth0	http, ssh, dns	有効
external	eth1	http, ssh, dns	無効

また、このサーバには、ファイアーウォールの他に IPS も導入する。IPS にはオープンソースソフトウェアの Snort を使用し、HTTP のパケット検査を行う。このため、HTTP のパケットは Snort へ転送するように firewalld の設定を行う。

5　システムの構築

前節で設計したシステムの構築手順を順に示す。

5.1　マスターイメージの作成

マスターイメージの作成手順を以下に示す。実際のコマンド操作については付録 A を参照されたい。

1. MicroSD カードに CentOS7 のインストールを行う。
2. パーティションの拡張を行う。
3. ソフトウェアのインストールを行う。

5.2　マスターイメージの複製

マスターイメージの作成に使用した Raspberry Pi 2 Model B から MicroSD カードを取り出し、このカードのクローンを 3 枚作成する。クローンの作成には、作業用のコンピュータが必要となる。作業用のコンピュータが Linux であれば、dd コマンドを使用してクローンの作成を行う。コンピュータが Windows であれば、ディスクイメージのバックアップとリカバリーが可能な任意のアプリケーションを使用する。

作成したクローンとマスターを合わせた 4 枚の MicroSD カードを、ファイアーウォール、DNS サーバ、ディレクトリサーバ、Web サーバに使用する Raspberry Pi 2 Model B へそれぞれ挿入する。また、マスターイメージの作成に使用した Raspberry Pi 2 Model B は、上記した 4 台のサーバのいずれかに転用する。

5.3　サーバの構築

以下に、サーバごとに行う構築手順を示す。ただし、サーバの構築はインターネットと通信可能な環境で行うものとする。

5.3.1　DNS サーバの構築

DNS サーバの構築手順を以下に示す。実際のコマンド操作については付録 B を参照されたい。

1. ビューの設定を行う。
2. 正引きゾーン及び逆引きゾーンを作成する。
3. IPv6 の無効化を行う。
4. 名前解決の順序を変更する。
5. ファイアーウォールの設定を行う。
6. NIC の設定を行う。
7. BIND の起動確認を行う。

5.3.2　ディレクトリサーバの構築

ディレクトリサーバの構築を行う。構築の手順については***の報告書を参照されたい。

5.3.3　Web サーバの構築

Web サーバの構築手順を以下に示す。実際のコマンド操作については付録 C を参照されたい。

1. 脆弱サイトを作成する。
2. MariaDB の起動を行う。
3. データベースを作成する。

4. ファイアーウォールの設定確認を行う。
5. NIC の設定を行う。
6. Apache の起動確認を行う。

5.3.4 ファイアーウォールの構築 (Snort の設定)

ファイアーウォールの構築は、Snort の設定、ネットワークの設定、ファイアーウォールサービスの設定に大別される。このうち、ネットワークとファイアーウォールサービスの設定を行うためには、ファイアーウォールに使用している Raspberry Pi 2 Model B が 2 つのネットワークに接続されている環境を要する。よって、現段階では Snort の設定のみを行い、その他の設定はシステムの結合時に行う。

以下は、Snort の設定手順である。実際のコマンド操作については付録 D.1 を参照されたい。

1. ルールファイルのダウンロードを行う。
2. ルールファイルの配置を行う。
3. ローカルルールの設定を行う。
4. ブラックリストファイルとホワイトリストファイルの作成を行う。
5. Snort の動作設定を行う。
6. グループとユーザの作成を行う。
7. パーミッションの変更を行う。

この実装段階では、Snort は動作しないため、付録 D.1.9 に示した起動確認は行わない。

5.4 システムの結合

設計したネットワーク構成に従い、ルータとスイッチ及びこれまでに構築した 4 台のサーバを LAN ケーブルで接続する。そして、演習システム内部と外部の間で通信を実現するために、ルータとファイアーウォールの設定を行う。

5.4.1 ルータの設定

ルータの設定手順を以下に示す。実際のコマンド操作については付録 E を参照されたい。

1. 作業用コンピュータとルータの接続を行う。
2. インタフェースに IP アドレスを設定する。
3. ルーティングの設定を行う。
4. 設定の確認を行う。
5. コンフィグレーションの保存を行う。

5.4.2 ファイアーウォールの構築 (ネットワークとファイアーウォールサービスの設定)

ルータとスイッチの電源が投入されている状態で、ネットワークの設定とファイアーウォールサービスの設定を行う。ネットワーク機器の電源を投入しておく理由は、ファイアーウォールが持つ 2 つの NIC を有効にするためである。

まず、以下に示す手順に従って、ネットワークを設定する。実際のコマンド操作については付録 D.2 を参照されたい。

1. NIC の設定を行う。
2. ルーティングの確認を行う。

次に、以下に示す手順に従って、ファイアーウォールサービスを設定する。実際のコマンド操作については付録 D.3 を参照されたい。

1. ゾーンの設定を行う。
2. パケットの転送設定を行う。
3. 設定の反映と確認を行う。

最後に、設定した Snort の動作確認を付録 D.1.9 を参考に実行する。

6 検証実験

本節では、構築したサイバーセキュリティ演習システムが設計通りに動作するかどうかを検証する。

6.1 実験方法

以下に示す手順に従って、演習システムの動作確認を行う。

1. 動作確認を行うためのクライアントを演習システムのインターネット側に用意する。クライアントのデフォルトゲートウェイには、ルータの Fa0 に設定した IP アドレスを指定する。また、クライアントが使用する DNS サーバには、演習システム内の DNS サーバを指定する。
2. 演習システムに電源を投入する。Snort はファイアーウォール起動時に自動起動しないため、手動にて起動しておく。
3. クライアント上でブラウザを起動し、脆弱サイト (`http://ysato.net/exploit_site/`) にアクセスする。これにより、DNS サーバによる名前解決、Web サーバの動作、ファイアーウォールの設定とルーティング、ルータのルーティングが行われていることを確認する。
4. 脆弱サイトのログインフォームで、Name に「hogeo」、Password に「hoge」と入力し、ログインを実行するために「Sign In」ボタンを押す。これにより、ディレクトリサーバの動作を確認する。
5. 会員検索フォームで、Name に「hogeo」を入力し、「検索」ボタンを押下して検索を行う。検索結果に、Name が「hogeo」のレコードが表示されていることを確認する。これにより、データベースの動作を確認する。
6. ブラウザの戻る機能を使用し、会員検索フォームに戻る。Name に「' OR '1'='1」を入力して、「検索」ボタンを押す。検索結果では、レコードが 5 件表示されていることを確認する。これにより、脆弱サイトの SQL インジェクション脆弱性を確認する。
7. 会員検索フォームに戻る。Name に「<script>alert(”XSS”)</script>」を入力して、「検索」ボタンを押す。画面遷移後、ブラウザにアラートダイアログが表示されることを確認する。これにより、脆弱サイトの XSS 脆弱性を確認する。確認後、「OK」ボタンを押してアラートダイアログを閉じる。
8. 会員検索フォームに戻る。ブラウザ上でタブを新規作成し、`http://ysato.net/exploit_site/logout.php` にアクセスする。その後、会員検索を実行していたタブでページを更新すると、ログイン画面に遷移することを確認する。これにより、脆弱サイトの CSRF 脆弱性を確認する。
9. SSH クライアントを起動し、ファイアーウォール (192.168.2.253) に SSH 接続する。ユーザ名は「root」であり、パスワードは「centos」である。SSH 接続後、コマンドラインに「tail /var/log/snort/alert」と入力し、クライアントと Web サーバ間の通信が IPS によって検知されていることを確認する。

6.2 実験環境

演習システムのインターネット側に、手順を実行するためのクライアントを用意した。表 9 にクライアントのシステム環境を示す。また、表 10 にクライアントのネットワーク設定を示す。

表 9　クライアントのシステム環境

役割	ソフトウェア
OS	Windows 10 Pro (OS ビルド 10586.104)
ブラウザ	Microsoft Edge 25.10586.0.0
SSH クライアント	Tera Term Version 4.79

表 10　クライアントのネットワーク設定

項目	設定値
IP アドレス	192.168.0.90
デフォルトゲートウェイ	192.168.0.2
DNS サーバ	192.168.2.2

6.3　実験結果

`http://ysato.net/exploit_site/` にアクセスしたときのブラウザ表示結果を図 5 に示す。また、Name に「hogeo」、Password に「hoge」と入力し、ログインを実行した後の表示結果を図 6 に示す。

図 5　`http://ysato.net/exploit_site/` にアクセスしたときの画面

図 6　ログイン後の画面

図 5 より、ドメイン名を用いて脆弱サイトにアクセスできたことが分かる。さらに、脆弱サイトにログインできたことが図 6 より分かる。

図 6 のページにおいて、Name に「hogeo」を入力したときの検索結果を図 7 に示す。また、Name に「' OR '1'='1」を入力したときの検索結果を図 8 に示す。

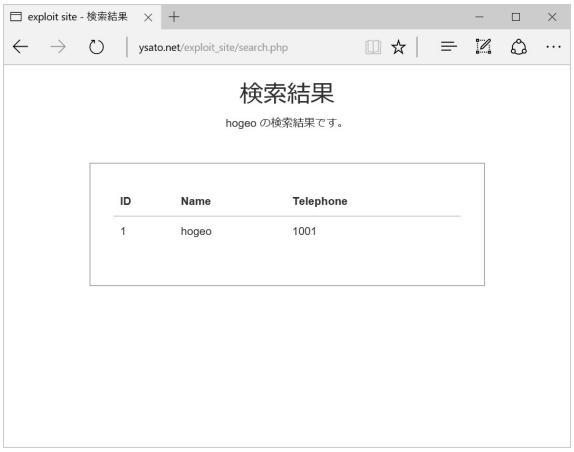

図 7　hogeo の検索画面

図 8　' OR '1'='1 の検索画面

図 7 より、検索が成功したことが分かる。さらに、図 8 より、SQL インジェクション攻撃が成功したことが分かる。

図 6 のページにおいて、Name に「<script>alert(”XSS”)</script>」を入力したときの検索結果を図 9 に示す。また、新しいタブで `http://ysato.net/exploit_site/logout.php` にアクセスした後、図 6 のページを更新したときの表示結果を図 10 に示す。

図 9　<script>alert(”XSS”)</script> の検索画面

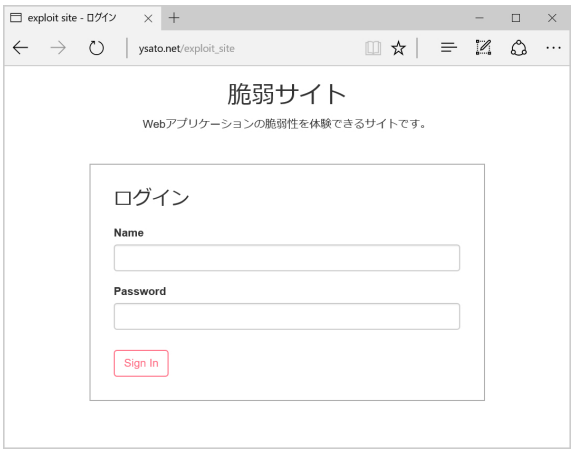

図 10　ページ更新後の画面

図 9 より、XSS 攻撃が成功したことが分かる。また、図 10 より、CSRF 攻撃が成功したことが分かる。

ファイアーウォールで tail /var/log/snort/alert コマンドを実行した結果をリスト 1 に示す。

リスト 1　tail /var/log/snort/alert の実行結果

```
[root@rpi2 ~]# tail /var/log/snort/alert
01/02-12:22:49.219971  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62079 -> 192.168.2.4:80
01/02-12:23:02.900935  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62129 -> 192.168.2.4:80
01/02-12:23:02.926768  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62130 -> 192.168.2.4:80
01/02-12:23:03.000959  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62132 -> 192.168.2.4:80
01/02-12:23:16.665513  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62180 -> 192.168.2.4:80
01/02-12:23:21.485153  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62191 -> 192.168.2.4:80
```

```
01/02-12:25:36.819134  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62298 -> 192.168.2.4:80
01/02-12:25:44.667996  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62306 -> 192.168.2.4:80
01/02-12:25:58.381064  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62328 -> 192.168.2.4:80
01/02-12:26:02.218114  [**] [1:1000027:1] HTTP/80 Rule [**] [Priority: 0] {TCP}
    192.168.0.90:62334 -> 192.168.2.4:80
```

リスト 1 より、1 月 2 日の 12 時 20 分頃に 192.168.0.90 から 192.168.2.4 への HTTP 通信が行われていたことが分かる。

7 おわりに

本稿では、サイバーセキュリティの実践的な演習が可能なシステムを設計し、構築の手順について示した。また、構築したシステムを使用し、Web アプリケーションに対する脆弱性攻撃を実行することで、システムの具体的な使用例を提示した。

今後の課題は、IPS の機能を活用することである。まず、Snort には正規表現を用いてパケットのパターンマッチングを行う機能が存在するが、本稿では機能させることができなかった。これが機能するようになると、システムを攻撃するだけでなく、システムを保護するためのセキュリティ演習も行うことができるようになる。また、SIEM などを導入し、Snort が蓄積するログの視覚的な分析やパケットのリアルタイムなレポーティングができるようになると、より実践的なインシデント対応やディジタルフォレンジックなどの演習が可能になる。以上の点は、今後の課題としたい。

参考文献

[1] イントラネットを利用するための運用上のセキュリティ対策, https://www.ipa.go.jp/security/fy18/reports/contents/enterprise/1.pdf (参照 2016-02-23).

付録 A　マスターイメージの作成

マスターイメージを作成するための手順を以下に示す。

A.1　CentOS7 のインストール

まず、Raspberry Pi 2 用の CentOS7 イメージ (CentOS-Userland-7-armv7hl-Minimal-1511-RaspberryPi2.img.xz) を http://mirror.centos.org/altarch/7/isos/armhfp/ よりダウンロードする。次に、ダウンロードしたファイルからイメージファイルを展開し、MicroSD カードに書き込む。書き込みが終わったら、Raspberry Pi 2 Model B に MicroSD カードを挿入し、CentOS7 の起動を確認する。ユーザ名の初期設定は「root」であり、パスワードの初期設定は「centos」である。

A.2　パーティションの拡張

パーティションを拡張するために、空の rootfs-repartition ファイルを作成し、再起動する。

```
[root@rpi2 ~]# touch /.rootfs-repartition
[root@rpi2 ~]# reboot
```

再起動後、df コマンドによりパーティションが拡張されていることを確認する。

```
[root@rpi2 ~]# df -H
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        31G  692M   29G   3% /
devtmpfs        482M     0  482M   0% /dev
tmpfs           486M     0  486M   0% /dev/shm
tmpfs           486M  6.4M  480M   2% /run
tmpfs           486M     0  486M   0% /sys/fs/cgroup
/dev/mmcblk0p1  315M   67M  248M  22% /boot
tmpfs            98M     0   98M   0% /run/user/0
```

A.3　ソフトウェアのインストール

yum を使用して、マスターイメージにソフトウェアのインストールを行う。しかし、Snort と DAQ は ARMv7 向けパッケージが存在しないため、手動にてコンパイルとインストールを行う。

また、以下の手順はインターネットと通信可能な環境で行うものとする。ネットワークの設定は、インターネットに繋がった LAN ケーブルを Raspberry Pi 2 Model B の LAN ポートに挿入したときに自動で行われるため、特に操作をする必要はない。

A.3.1　yum によるインストール

まず、インストールされているソフトウェアのアップデートを行う。その後、ソフトウェアのインストールを行う。

```
[root@rpi2 ~]# yum update
[root@rpi2 ~]# yum groupinstall "Development Tools" --skip-broken
[root@rpi2 ~]# yum install httpd mariadb mariadb-server php-mysql bind bind-utils
    openldap-servers openldap-clients php-ldap libpcap libpcap-devel pcre pcre-devel
    libdnet libdnet-devel libnfnetlink libnfnetlink-devel libmnl libmnl-devel
    libnetfilter_queue libnetfilter_queue-devel emacs vim wget ntp maek zlib zlib-devel
     java-1.8.0-openjdk java-1.8.0-openjdk-devel perl php ruby
```

A.3.2 DAQ と Snort のインストール

DAQ と Snort をインストールする前に、CentOS7 上の時刻を設定する。時刻が正しくない場合、DAQ と Snort のコンパイルに失敗する。時刻の設定手順を次に示す。

```
[root@rpi2 ~]# cp -p /usr/share/zoneinfo/Japan /etc/localtime
[root@rpi2 ~]# ntpdate ntp.nict.jp
```

DAQ のインストール手順を次に示す。

```
[root@rpi2 ~]# wget https://www.snort.org/downloads/snort/daq-2.0.6.tar.gz
[root@rpi2 ~]# tar xvfz daq-2.0.6.tar.gz
[root@rpi2 ~]# cd daq-2.0.6/
[root@rpi2 daq-2.0.6]# ./configure

# 以下の出力を確認
# Build NFQ DAQ module....... : yes

[root@rpi2 daq-2.0.6]# make
[root@rpi2 daq-2.0.6]# make install
```

Snort のインストール手順を次に示す。

```
[root@rpi2 ~]# wget https://www.snort.org/downloads/snort/snort-2.9.8.0.tar.gz
[root@rpi2 ~]# tar xvfz snort-2.9.8.0.tar.gz
[root@rpi2 ~]# cd snort-2.9.8.0/
[root@rpi2 snort-2.9.8.0]# ./configure
[root@rpi2 snort-2.9.8.0]# make
[root@rpi2 snort-2.9.8.0]# make install
```

付録 B　DNS サーバの実装

DNS サーバを構築するための設定手順を以下に示す。

B.1　ビューの設定

BIND の設定ファイル/etc/named.conf の編集を行う。編集を行った箇所は、opntions セクション内の先頭 2 行と、logging セクション以降である。

```
[root@rpi2 ~]# vi /etc/named.conf
options {
        listen-on port 53 { any; };
        // listen-on-v6 port 53 { ::1; };
        directory       "/var/named";
        dump-file       "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        allow-query { any; };
        recursion yes;

        dnssec-enable yes;
        dnssec-validation yes;
        dnssec-lookaside auto;

        /* Path to ISC DLV key */
        bindkeys-file "/etc/named.iscdlv.key";

        managed-keys-directory "/var/named/dynamic";

        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
};

logging {
        channel default_debug {
                file "data/named.run";
                severity dynamic;
        };
};

view "external" {
        match-clients { any; };
        recursion no;
        allow-transfer { none; };

        zone "." IN {
                type hint;
                file "named.ca";
        };

        include "/etc/named.rfc1912.zones";
        include "/etc/named.root.key";

        zone "ysato.net" {
                type master;
```

```
                file "ysato.net.wan";
                allow-update { none; };
        };

        zone "2.168.192.in-addr.arpa" {
                type master;
                file "2.168.192.in-addr.arpa.db";
                allow-update { none; };
        };
};
```

B.2 正引きゾーンの作成

ホスト名を IP アドレスに変換するための正引きゾーン/var/named/ysato.net.wan を作成する。

```
[root@rpi2 ~]# vi /var/named/ysato.net.wan
$TTL 86400
@ IN      SOA  ysato.net. root.ysato.net. (
                      2015111302 ; serial
                      3600       ; refresh 1hr
                      900        ; retry 15min
                      604800     ; expire 1w
                      86400      ; min 24hr
)
      IN  NS  dns.ysato.net.
      IN  A 192.168.2.4
dns   IN  A 192.168.2.2
ldap  IN  A 192.168.2.3
www   IN  CNAME ysato.net.
```

B.3 逆引きゾーンの作成

IP アドレスをホスト名に変換するための逆引きゾーン/var/named/2.168.192.in-addr.arpa.db を作成する。

```
[root@rpi2 ~]# vi /var/named/2.168.192.in-addr.arpa.db
$TTL 86400
@ IN      SOA     ysato.net. root.ysato.net.  (
                     2015111303      ; Serial
                     3600            ; Refresh
                     900             ; Retry
                     604800          ; Expire
                     3600 )          ; Minimum
  IN  NS  dns.ysato.net.
2 IN  PTR dns.ysato.net.
3 IN  PTR ldap.ysato.net.
4 IN  PTR www.ysato.net.
```

B.4 IPv6 の無効化

IPv6 を無効にするために、/etc/hosts の IPv6 に関する行をコメントアウトする。

```
[root@rpi2 ~]# vi /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
#::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
```

また、/etc/sysconfig/named を編集し、BIND が IPv4 専用モードで起動するように設定する。

```
[root@rpi2 ~]# vi /etc/sysconfig/named
OPTIONS="-4"
```

B.5 順序設定ファイルの変更

名前解決を hosts、dns の順で行うために、次に示すように編集する。

```
[root@rpi2 ~]# vi /etc/host.conf
order hosts,bind
```

また、/etc/nsswitch.conf を編集し、hosts の設定を次のように変更する。

```
[root@rpi2 ~]# vi /etc/nsswitch.conf
(略)
#hosts:     db files nisplus nis dns
hosts:      files dns myhostname
```

B.6 ファイアウォールの設定

DNS サーバ上のファイアーウォールに DNS サービスを許可させる。

```
[root@rpi2 ~]# firewall-cmd --add-service=dns
[root@rpi2 ~]# firewall-cmd --add-service=dns --permanent
```

B.7 NIC の設定

nmtui コマンドを入力し、NetworkManager を起動する。Edit a connection を選択し、有効な NIC の設定を図 11 のように変更する。設定後は、NetworkManager を再起動し、設定を反映させる。

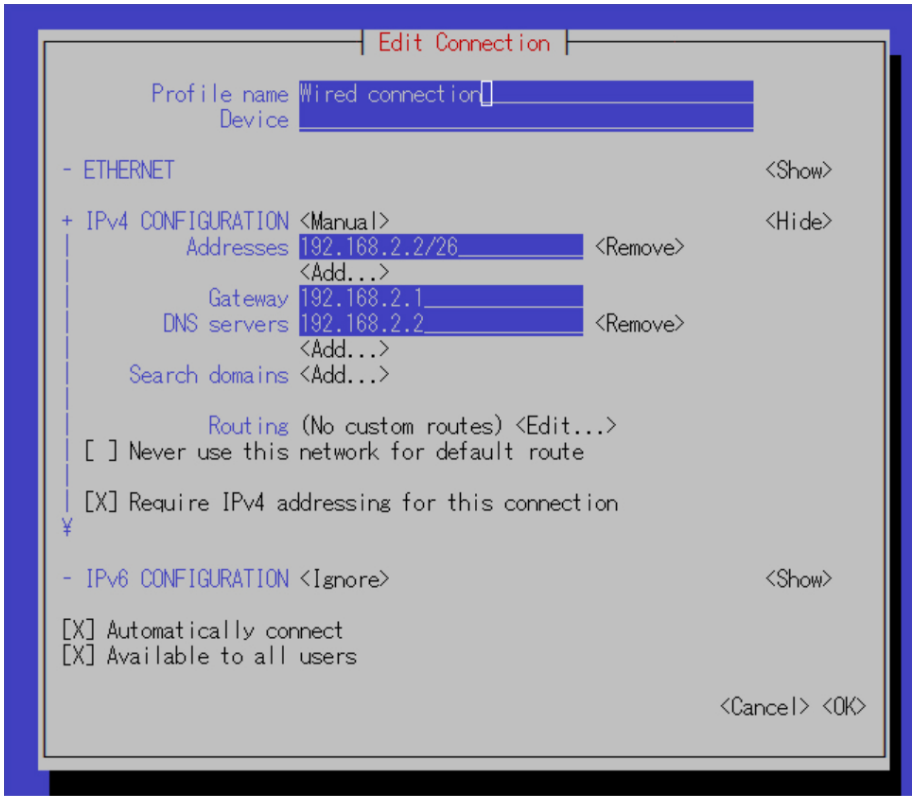

図 11　NIC の設定

B.8　BIND の起動

BIND の起動を確認する。また、再起動後に BIND が自動起動するように設定する。

```
[root@rpi2 ~]# systemctl start named
[root@rpi2 ~]# systemctl enable named
```

付録 C　Web サーバの実装

Web サーバを構築するための設定手順を以下に示す。

C.1　脆弱サイトの作成

/var/www/html/に exploit_site ディレクトリを作成し、脆弱サイトで使用する CSS ライブラリを配置する。

```
[root@rpi2 ~]# mkdir /var/www/html/exploit_site/
[root@rpi2 ~]# cd /var/www/html/exploit_site/
[root@rpi2 exploit_site]# wget https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/
    bootstrap.min.css
```

作成した exploit_site ディレクトリ内に、リスト 2-8 に示す脆弱サイトのソースファイルを作成する。

リスト 2　index.php

```
<?php
ini_set("display_errors", Off);
error_reporting(E_ALL);

require "database.php";
require "ldapserver.php";
$errorMessage;

session_start();

if (isset($_POST["login"]))
{
    try {
        onLogInClick();
    }
    catch(Exception $e){
        $errorMessage = $e->getMessage();
    }
}

// ログインボタンがクリックされたら検証する
function onLogInClick()
{
    $auth = isldapAuth($_POST["name"],$_POST["pass"]);

    if($auth)
    {
        if (chkGroup($auth, $_POST["name"]))
        {
            session_regenerate_id(true);
            $_SESSION["USERID"] = $_POST["name"];
            header("Location: main.php");
        }
        else
        {
            throw new Exception("アクセス権限がありません。");
        }
    }
```

```
    else
    {
        throw new Exception("NameまたはPasswordに誤りがあります。");
    }
}
?>

<!DOCTYPE html>
<html>
    <head>
        <title>exploit site - ログイン</title>

        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <!-- Bootstrap -->
        <link rel="stylesheet" href="./bootstrap.min.css">

        <link rel="stylesheet" href="./main.css" />
    </head>
    <body>
        <div class="contents">
            <div id="abstract">
                <h2>脆弱サイト</h2>
                <p>Webアプリケーションの脆弱性を体験できるサイトです。 </p>
            </div>

            <div id="main_content">
                <h3>ログイン</h3>
                <?php if (!empty($errorMessage)) { ?>
                <div class="alert alert-danger" role="alert">
                    <strong style="margin-right: 1em;">エラー</strong>
                    <?php echo $errorMessage;?>
                </div>
                <?php } ?>

                <form method="post" action="./index.php">
                    <div class="form-group">
                        <label for="input-name">Name</label>
                        <input required type="text" class="form-control" id="input-
                            name" name="name" placeholder="Enter name">
                    </div>
                    <div class="form-group">
                        <label for="input-pass">Password</label>
                        <input required type="text" class="form-control" id="input-
                            pass" name="pass" placeholder="Enter password">
                    </div>
                    <input type="submit" class="btn btn-default" id="submit-button"
                        name="login" value="Sign In">
                </form>
            </div>
        </div>
    </body>
</html>
```

リスト 3　ldapserver.php

```
<?php
$base['dn'] = 'dc=my-ldap,dc=server';
$base['host'] = '192.168.2.3';
$base['port'] = 389;
$base['people'] = 'ou=People,'.$base['dn'];
$base['group'] = 'ou=Group,'.$base['dn'];
$base['admgr'] = 'cn=admin,'.$base['group'];
$base['stfgr'] = 'cn=staff,'.$base['group'];

function isldapAuth ($name,$pass) {

    global $base;

    $link_id = ldap_connect($base['host'],$base['port']);

    if ($link_id) {
        $ldapdn = 'uid='.$name.','.$base['people'];
        $attrs = array('memberof');
        ldap_set_option($link_id, LDAP_OPT_PROTOCOL_VERSION, 3);
        ldap_set_option($link_id, LDAP_OPT_REFERRALS, 0);

        $bind = ldap_bind($link_id,$ldapdn,$pass);
        if ($bind) {
            return $link_id;
        }
    }
    else {
        return false;
    }
}


function chkGroup ($link_id, $name) {

    global $base;
    $filt = '(& (uid='.$name.') (| (memberof='.$base['admgr'].')(memberof='.$base["
        stfgr"].')))';

    $res = ldap_search($link_id, $base['people'], $filt);
    $info = ldap_get_entries($link_id,$res);
    print_r($info);
    if ($info['count']) {
        return $link_id;
    }
    else {
        return false;
    }
}
?>
```

リスト 4　main.php

```
<?php
session_start();
```

```
// ログイン状態のチェック
if (!isset($_SESSION["USERID"]))
{
  header("Location: logout.php");
  exit;
}
?>

<!DOCTYPE html>
<html>
    <head>
        <title>exploit site - メイン</title>

        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <!-- Bootstrap -->
        <link rel="stylesheet" href="./bootstrap.min.css">

        <link rel="stylesheet" href="./main.css" />
    </head>
    <body>
        <div class="contents">
            <div id="abstract">
                <h2>ようこそ</h2>
                <p>このページでは会員検索が行えます。</p>
            </div>

            <div id="main_content">
                <h3>会員検索</h3>
                <form class="input-group" method="post" action="./search.php">
                    <label for="input-name">Name</label>
                    <input required type="text" class="form-control" id="input-name"
                        name="name" placeholder="Enter name">
                    <span class="input-group-btn">
                        <button type="submit" style="margin: 0; margin-top: 25px"
                            class="btn btn-default" name="search" value="Search">
                            検索
                        </button>
                    </span>
                </form>
            </div>

            <div style="margin: auto; width: 500px; margin-top: 20px;">
                <form action="./logout.php" method="GET" style="margin-top:20px;">
                    <input type="submit" class="btn btn-default" id="submit-button"
                        value="Sign Out"/>
                </form>
            </div>
        </div>
    </body>
</html>
```

リスト 5　search.php

```
<?php
require "database.php";
session_start();

// ログイン状態のチェック
if (!isset($_SESSION["USERID"]))
{
    header("Location: logout.php");
    exit;
}

$tbody_str = "";

$result = selectFromUser("name = '".$_POST["name"]."'");
while ($row = $result->fetch_assoc())
{
    $tbody_str .= "<tr>" .td($row["id"]) .td($row["name"]) .td($row["tel"]) ."</tr>";
}

function td($value)
{
    return "<td>".$value."</td>";
}
?>

<!DOCTYPE html>
<html>
    <head>
        <title>exploit site - 検索結果</title>

        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <!-- Bootstrap -->
        <link rel="stylesheet" href="./bootstrap.min.css">

        <link rel="stylesheet" href="./main.css" />
    </head>
    <body>
        <div class="contents">
            <div id="abstract">
                <h2>検索結果</h2>
                <p><?php echo $_POST["name"]?> の検索結果です。</p>
            </div>

            <div id="main_content">
                <?php if (empty($tbody_str)) { ?>
                <div class="alert alert-danger" role="alert">
                    該当する結果がありませんでした。
                </div>
                <?php } ?>

                <table class="table">
                    <thead>
                        <tr>
```

```
                            <th>ID</th>
                            <th>Name</th>
                <th>Telephone</th>
                        </tr>
                    </thead>
                    <tbody>
                        <?php echo $tbody_str; ?>
                    </tbody>
                </table>
            </div>
        </div>
    </body>
</html>
```

リスト 6　database.php

```
<?php
// DB接続設定
$db['host'] = "localhost";
$db['user'] = "root";
$db['pass'] = "";
$db['dbname'] = "web";

//usage predicate: name = 'hogeo'
function selectFromUser($predicate)
{
    global $db;
    $mysqli = new mysqli($db['host'], $db['user'], $db['pass'], $db['dbname']);

    $result;
    if ($predicate == "")
    {
        $result = selectAll($mysqli);
    }
    else
    {
        $result = selectUserWhereName($mysqli, $predicate);
    }
    $mysqli->close();
    return $result;
}

function selectAll($mysqli)
{
    return $mysqli->query("SELECT * FROM user;");
}

function selectUserWhereName($mysqli, $predicate)
{
  return $mysqli->query("SELECT * FROM user WHERE " .$predicate .";");
}
?>
```

リスト 7　logout.php

```
<?php
session_start();

$_SESSION = array();
@session_destroy();

header("Location: index.php");
?>
```

リスト 8　main.css

```
.contents {
  margin: 20px;
}

#submit-button {
    color: #FF7187;
    border-color: #FF7187;
    margin-top: 10px;
}

#abstract {
  text-align: center;
  margin: auto;
    margin-bottom: 40px;
}

#main_content {
  margin: auto;
    width: 500px;
    padding: 30px;
    border-color: #AFAFAF;
    border-width: 1px;
    border-style: solid;
}
#main_content h3 {
    margin-top: 0;
    margin-bottom: 20px;
}
```

作成したソースコードのパーミッションを設定する。

```
[root@rpi2 ~]# chmod 705 /var/www/html/exploit_site/*
[root@rpi2 ~]# ls -al /var/www/html/exploit_site/
total 148
drwxr-xr-x 2 root root   4096 Jan  1 09:04 .
drwxr-xr-x 3 root root   4096 Feb 21  2016 ..
-rwx---r-x 1 root root 114011 Feb 21  2016 bootstrap.min.css
-rwx---r-x 1 root root    898 Feb 21  2016 database.php
-rwx---r-x 1 root root   2980 Feb 21  2016 index.php
-rwx---r-x 1 root root   1090 Feb 21  2016 ldapserver.php
-rwx---r-x 1 root root    216 Feb 21  2016 logout.php
-rwx---r-x 1 root root    408 Feb 21  2016 main.css
-rwx---r-x 1 root root   1825 Feb 21  2016 main.php
-rwx---r-x 1 root root   1753 Feb 21  2016 search.php
```

C.2 MariaDB の起動

MariaDB を起動する。また、再起動後に自動起動するように設定する。

```
[root@rpi2 ~]# systemctl start mariadb
[root@rpi2 ~]# systemctl enable mariadb
```

C.3 データベースの作成

MariaDB に web データベースを作成し、Web アプリケーションで使用するテーブルとレコードを定義する。

```
[root@rpi2 ~]# mysql -u root
MariaDB [(none)]> CREATE DATABASE web;

MariaDB [(none)]> USE web;

MariaDB [(none)]> CREATE TABLE `user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `tel` varchar(45) NOT NULL,
  `pass` varchar(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `id_UNIQUE` (`id`),
  UNIQUE KEY `name_UNIQUE` (`name`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;

MariaDB [(none)]> INSERT INTO `user` VALUES
(1,'hogeo','1001','hoge'),
(2,'pugeo','1012','puge'),
(3,'taro','1008','taro'),
(4,'fooko','1003','foo'),
(5,'barko','1005','bar');

MariaDB [(none)]> exit
```

C.4 ファイアウォールの設定確認

ファイアーウォールの設定を確認し、http が許可されていることを確認する。

```
[root@rpi2 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: eth0
  sources:
  services: dhcpv6-client http https ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

C.5　NIC の設定

nmtui コマンドを入力し、NetworkManager を起動する。Edit a connection を選択し、有効な NIC の設定を図 12 のように変更する。設定後は、NetworkManager を再起動し、設定を反映させる。

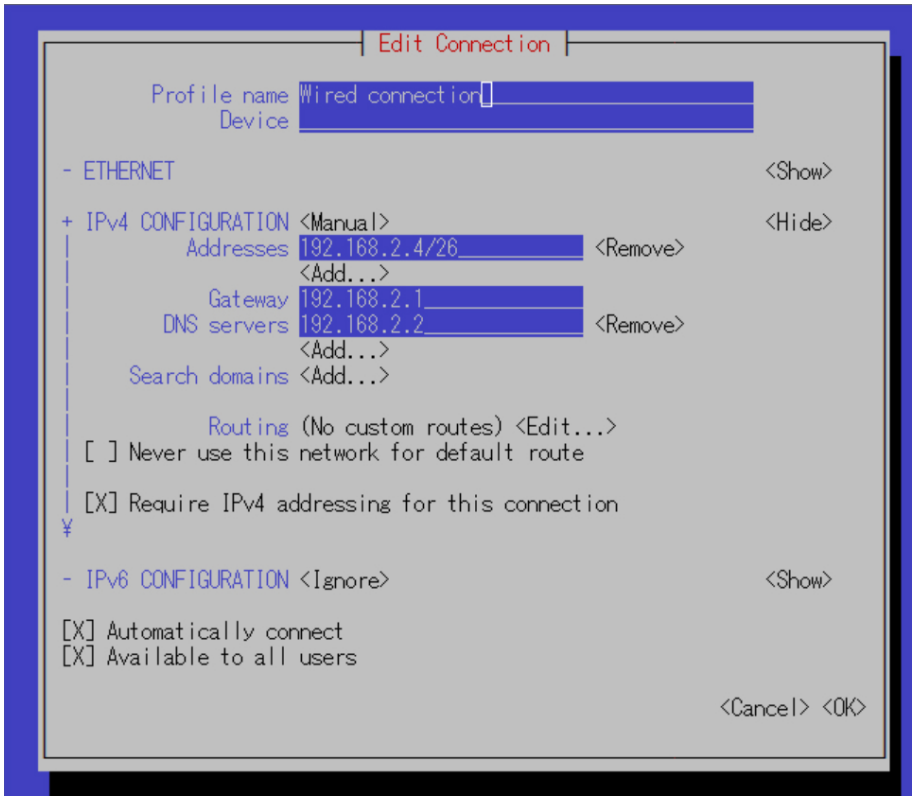

図 12　NIC の設定

C.6　Apache の起動

Apache を起動する。また、再起動後に自動起動するように設定する。

```
[root@rpi2 ~]# systemctl start httpd
[root@rpi2 ~]# systemctl enable httpd
```

付録 D　ファイアーウォールの実装

ファイアーウォールを構築するための設定手順を以下に示す。

D.1　Snort

D.1.1　ルールファイルのダウンロード

Snort 公式サイト[1)]よりルールファイル（Registerd rules）をダウンロードし、ホームディレクトリに配置する。ただし、Registerd rules ファイルをダウンロードするためには、Snort 公式サイトからアカウント登録してログインする必要がある。

D.1.2　ルールファイルの配置

ダウンロードしたルールファイルを snortrules-snapshot-2972.tar.gz としたときの手順を次に示す。

```
[root@rpi2 ~]# mkdir /etc/snort
[root@rpi2 ~]# cd /etc/snort/
[root@rpi2 snort]# tar zxvf ~/snortrules-snapshot-2972.tar.gz
[root@rpi2 snort]# cp ./etc/* .
[root@rpi2 snort]# cp ~/snort-2.9.8.0/etc/* .
```

D.1.3　ダイナミックルールファイルの配置

```
[root@rpi2 ~]# mkdir /usr/local/lib/snort_dynamicrules
[root@rpi2 ~]# cp /etc/snort/so_rules/precompiled/RHEL-6-0/i386/2.9.7.6/* /usr/local/
    lib/snort_dynamicrules/
```

D.1.4　ローカルルールの記述

HTTP のパケットを検知するように、/etc/snort/rules/local.rules を編集する。

```
[root@rpi2 ~]# vi /etc/snort/rules/local.rules
（略）
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80 (msg:"HTTP/80 Rule"; sid:1000021; rev
    :1;)
```

ところで、行頭の「alert」を「drop」に書き換えれば、Snort は検知したパケットを破棄するように動作する。

D.1.5　空のブラックリストファイル及びホワイトリストファイルの作成

```
[root@rpi2 ~]# touch /etc/snort/rules/white_list.rules /etc/snort/rules/black_list.
    rules
```

D.1.6　Snort の設定

/etc/snort/snort.conf を次のように編集する。

[1)] https://www.snort.org/

```
[root@rpi2 ~]# vi /etc/snort/snort.conf
#ipvar HOME_NET any
ipvar HOME_NET [192.168.2.253/32, 192.168.2.0/26]

#ipvar EXTERNAL_NET any
ipvar EXTERNAL_NET !$HOME_NET

（略）

#var RULE_PATH ../rules
#var SO_RULE_PATH ../so_rules
#var PREPROC_RULE_PATH ../preproc_rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

（略）

#var WHITE_LIST_PATH ../rules
#var BLACK_LIST_PATH ../rules
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules

（略）

config daq: nfq
config daq_dir: /usr/local/lib/daq
config daq_mode: inline
#config daq_var: <type >
config policy_mode: inline

（略）

#dynamicdetection directory /usr/local/snort/lib/snort_dynamicrules
```

D.1.7 グループ及びユーザの作成

```
[root@rpi2 ~]# groupadd -g 511 snort
[root@rpi2 ~]# useradd snort -u 511 -d /var/log/snort -s /sbin/nologin -c 'Snort User'
     -g snort
[root@rpi2 ~]# chown -R snort.snort /etc/snort
[root@rpi2 ~]# chown -R snort.snort /var/log/snort
```

D.1.8 パーミッションの変更

```
[root@rpi2 ~]# cd /usr/local/lib/
[root@rpi2 lib]# chown -R snort.snort snort* pkgconfig
[root@rpi2 lib]# chmod -R 700 snort* pkgconfig
[root@rpi2 lib]# cd /usr/local/bin/
[root@rpi2 bin]# chown snort.snort daq-modules-config u2*
[root@rpi2 bin]# chmod 700 daq-modules-config u2*
```

D.1.9 Snort の起動

Snort の起動を確認する。

```
[root@rpi2 ~]# snort -Q --daq nfq --daq-var queue=2 -c /etc/snort/snort.conf -l /var/
    log/snort -A fast
（略）
Commencing packet processing (pid=15635)
Decoding Raw IP4
```

Decoding Raw IP4 の文字が出力された後、Snort はパケットを検知し続ける。Snort を終了する場合は、Ctrl + C を入力すれば良い。また、デーモンとして起動する場合には、Snort の起動オプションに-D を追加すれば良い。

D.2　ネットワーク

D.2.1　NIC の設定

USB-LAN 変換コネクタを挿入している状態で、nmtui コマンドを入力し、NetworkManager を起動する。Edit a connection を選択し、eth0 と eth1 の設定を図 13 と図 14 のように行う。

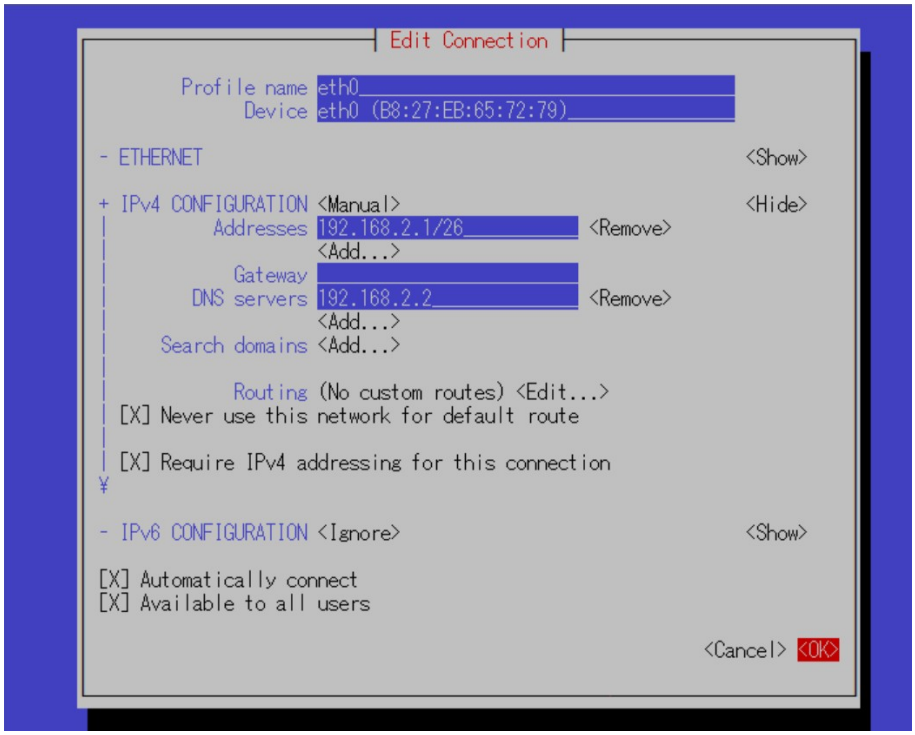

図 13　eth0 の設定

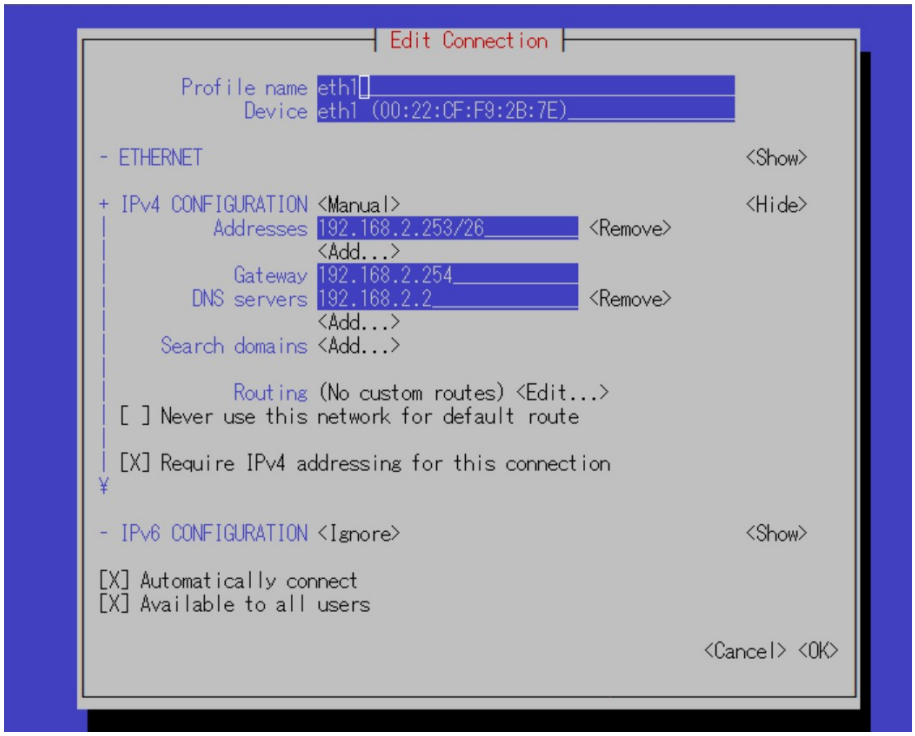

図 14　eth1 の設定

設定の反映と確認を行う。

```
[root@rpi2 ~]# systemctl restart NetworkManager
[root@rpi2 ~]# nmcli device show eth0
GENERAL.DEVICE:                         eth0
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         B8:27:EB:65:72:79
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     eth0
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/
    ActiveConnection/0
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         192.168.2.1/26
IP4.GATEWAY:
IP4.DNS[1]:                             192.168.2.2
IP6.ADDRESS[1]:                         fe80::ba27:ebff:fe65:7279/64
IP6.GATEWAY:

[root@rpi2 ~]# nmcli device show eth1
GENERAL.DEVICE:                         eth1
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         00:22:CF:F9:2B:7E
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     eth1
GENERAL.CON-PATH:                       /org/freedesktop/NetworkManager/
    ActiveConnection/1
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         192.168.2.253/26
IP4.GATEWAY:                            192.168.2.254
IP4.DNS[1]:                             192.168.2.2
IP6.ADDRESS[1]:                         fe80::222:cfff:fef9:2b7e/64
IP6.GATEWAY:
```

D.2.2 ルーティングの確認

NIC を設定すると、ルーティング設定が自動的に行われる。次に示すコマンドで、ルーティング設定の確認を行う。

```
[root@rpi2 ~]# netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         192.168.2.254   0.0.0.0         UG        0 0          0 eth1
192.168.2.0     0.0.0.0         255.255.255.192 U         0 0          0 eth0
192.168.2.0     0.0.0.0         255.255.255.192 U         0 0          0 eth0
192.168.2.192   0.0.0.0         255.255.255.192 U         0 0          0 eth1
192.168.2.192   0.0.0.0         255.255.255.192 U         0 0          0 eth1
```

この実行結果の意味を以下に示す。

- 192.168.2.0/26 上の IP アドレスへは eth0 を経由して直接接続する
- 192.168.2.192/26 上の IP アドレスへは eth1 を経由して直接接続する
- それ以外のネットワークにある IP アドレスへは eth1 を経由して 192.168.2.254 に接続する

D.3 ファイアーウォール

D.3.1 ゾーンの設定

external ゾーンと dmz ゾーンの設定を変更する。

```
インタフェースの変更
[root@rpi2 ~]# firewall-cmd --change-interface=eth1 --zone=external --permanent
[root@rpi2 ~]# firewall-cmd --change-interface=eth0 --zone=dmz --permanent
サービスの許可
[root@rpi2 ~]# firewall-cmd --add-service=http --zone=external --permanent
[root@rpi2 ~]# firewall-cmd --add-service=http --zone=dmz --permanent
[root@rpi2 ~]# firewall-cmd --add-service=dns --zone=external --permanent
[root@rpi2 ~]# firewall-cmd --add-service=dns --zone=dmz --permanent
IPマスカレードの有効化
[root@rpi2 ~]# firewall-cmd --add-masquerade --zone=dmz --permanent
```

D.3.2 パケットの転送設定

HTTP のパケットは Snort で検査するため、パケットの転送設定を行う。

```
[root@rpi2 ~]# firewall-cmd --direct --add-rule ipv4 nat PREROUTING 0 -i eth1 -d
    192.168.2.253 -p tcp --dport 80 -j DNAT --to 192.168.2.4:80 --permanent
[root@rpi2 ~]# firewall-cmd --direct --add-rule ipv4 filter FORWARD 0 -p tcp --dport
    80 -j NFQUEUE --queue-num 2 --permanent
```

D.3.3 設定の反映

firewalld を再起動して、設定を反映させる。

```
[root@rpi2 ~]# firewall-cmd --reload
```

再起動後、設定が反映されていることを確認する。

```
[root@rpi2 ~]# firewall-cmd --list-all --zone=external
external (active)
  interfaces: eth1
  sources:
  services: dns http ssh
  ports:
  masquerade: yes
  forward-ports:
  icmp-blocks:
  rich rules:

[root@rpi2 ~]# firewall-cmd --list-all --zone=dmz
dmz (active)
  interfaces: eth0
  sources:
  services: dns http ssh
  ports:
  masquerade: yes
  forward-ports:
  icmp-blocks:
```

```
  rich rules:

[root@rpi2 ~]# firewall-cmd --direct --get-all-rules
ipv4 nat PREROUTING 0 -i eth1 -d 192.168.2.253 -p tcp --dport 80 -j DNAT --to
    192.168.2.4:80
ipv4 filter FORWARD 0 -p tcp --dport 80 -j NFQUEUE --queue-num 2
```

付録 E　ルータの設定

ルータの設定には、ターミナル操作を行うためのコンピュータが必要である。以下では、ターミナルソフトウェアとして Tera Term Version 4.79 がインストールされた Windows 10 Pro (OS ビルド 10586.104) が動作するコンピュータを使用して設定を行う。

以下に、インターネットとイントラネットの境界に位置するルータの設定手順を示す。

E.1　接続

コンソールケーブルを用いて、ルータの CONSOLE ポートとコンピュータの USB を接続する。次に、Tera Term を起動し、「新しい接続」画面からシリアル接続を選択する (図 15)。

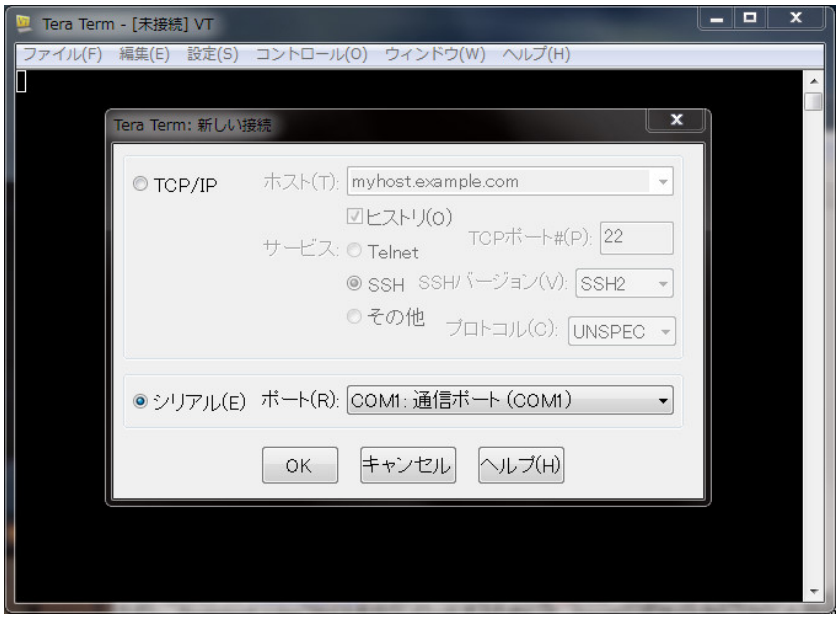

図 15　Tera Term によるルータのシリアル接続

ルータの電源を投入し、Cisco IOS の起動を待機する。起動後は、次に示すようにグローバルコンフィグレーションモードに移行する。

```
Router> enable
Router# configure terminal
Router(config)#
```

E.2　IP アドレスの設定

インターネット側の FastEthernet0 に 192.168.0.2/24 を、イントラネット側の FastEthernet1 に 192.168.2.254/26 を設定する。

```
Router(config)# interface FastEthernet0
Router(config-if)# ip address 192.168.0.2 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# interface FastEthernet1
Router(config-if)# ip address 192.168.2.254 255.255.255.192
Router(config-if)# no shutdown
Router(config-if)# exit
```

E.3 ルーティングの設定

宛先ネットワーク 192.168.2.0/26 に対するネクストホップアドレスとして 192.168.2.253 を設定する。

```
Router(config)# ip route 192.168.2.0 255.255.255.192 192.168.2.253
Router(config)# exit
```

E.4 設定の確認

show running-config コマンドを実行して設定の確認を行う。

```
Router# show running-config

 (略)

interface FastEthernet0
 ip address 192.168.0.2 255.255.255.0

interface FastEthernet1
 ip address 192.168.2.254 255.255.255.192

 (略)

ip route 192.168.2.0 255.255.255.192 192.168.2.253
```

E.5 コンフィグレーションの保存

現在の設定がルータ起動時に読み込まれるように設定する。

```
Router# copy running-config startup-config
```