一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
IA2020-20(2020-11)

# ブラウザ内で実現するネットワーク演習環境を用いたオンライン演習

飯田　勝吉†　　南　　弘征†

† 北海道大学　情報基盤センター
〒 060-0811 札幌市北区北 11 条西 5 丁目
E-mail: †{iida,min}@iic.hokudai.ac.jp

**あらまし**　新型コロナウィルス感染症の感染拡大に伴い、大学における様々な教育活動がオンライン化を余儀なくされている。特に、演習を伴う科目の場合にオンライン化が困難となる。本稿では、著者らが実施した UNIX を用いたネットワークログ解析演習のオンライン化について説明する。

**キーワード**　オンライン演習，ブラウザ，UNIX 演習.

## 1. はじめに

2020 年に生じた新型コロナウィルス感染症の感染拡大に伴い、多くの大学の講義がオンライン化されている。これまでにない経験していない事態の発生により、学生も教員も多くの困難に直面している。そのため、各大学ではオンライン講義に鳴れていない学生や教員に対する様々なサポートを実施している（北海道大学の例: [1]）。また、国立情報学研究所では、複数の大学で共通する課題とその対処方法を共有するべく「4 月からの大学等遠隔授業に関する取組状況共有サイバーシンポジウム」というオンライン講演会を実施している [2]。

オンライン化が困難な講義科目として演習や実習を伴う科目が存在する。これらの科目では、（１）実験設備のある場所でないと実施できない、（２）学生が演習作業中に困難に直面した際に教員やティーチングアシスタント (TA) からのサポートを受ける必要がある、（３）教員・TA の立場でも学生の立場でも演習作業中のサポートは対面の方が実施しやすい、（４）グループワークを伴う演習ではグループ内の学生間の密な連携が必要となる。このようなことから、演習や実習を伴う科目は実験設備のある特別な教室で、教員及び複数の TA のサポートのもと、対面で実施することが当然と考えられてきた。

本稿では、著者らが関与している文部科学省の教育事業（大学教育再生戦略推進費「成長分野を支える情報技術人材の育成拠点の形成（enPiT）」）[3] 内の演習科目をオンライン化し実施した内容を記載する。本演習科目はコンピュータやネットワークを使うセキュリティ技術に関する演習であり、その設備を整えた演習室で実施してきた。2020 年度は感染拡大を防止するため、オンライン化することとなった。オンライン化の方針は以下の通りである。

**オンライン化の方針：**

（１） 遠隔地から演習に参加できること。

（２） できるだけ過去の演習教材をそのまま利用できること。

（３） できるだけ多様な（学生の私物）PC 環境から利用できること。

（４） できるだけ複雑な手順を用いず、初心者でも簡便に演習に参加できること。

（５） 演習に利用する（学生の私物）PC のセキュリティ環境に影響を与えたり与えられたりしないこと。

（６） 演習環境の外部に対するセキュリティ攻撃を行えないこと。

（７） 学生のサーバ側の演習環境が学生ごとに分離されていること。

（８） 悪意のある第３者により演習用のサーバが踏み台にされないこと。

（９） 期限内にソフトウェアの実装、テスト、実習書の執筆などが完了すること (注1)。

上記の方針のもと、学生には以下のような受講環境の準備を要請した。（１）受講場所にインターネット接続された PC があること、（２）マイク、ヘッドフォンなどが PC に接続され Zoom による教員等の説明の聴講や教員・TA に対する音声による質問が可能であること、（３）PC には JavaScript が動くブラウザがインストールされていること。本演習環境の大きな特徴は、JavaScript が動くブラウザであれば、OS やブラウザの種類やバージョンに依存せず、多くの環境で履修できることである。

以下、２節で教育事業全体の概要と本演習の紹介を記載し、３節でブラウザ内で実現するオンライン演習環境の設計と実装を述べる。４節で実際に実施した演習の状況とその考察を述べ、５節でまとめる。

## 2. 教育事業全体の概要と本演習の紹介

文部科学省大学教育再生戦略推進費「成長分野を支える情報技術人材の育成拠点の形成（enPiT）」[3] は、情報技術を高度に活用して社会の具体的な課題を解決できる人材の育成機能を強化するため、産学協働の実践教育ネットワークを形成し、課題解決型学習（PBL）等の実践的な教育を推進し広く全国に普及させることを目的として、令和 2 年度まで予算措置が講じられている。うち、セキュリティ分野 [4] について、大学院修士課程を対象として平成 22 年度から実施された第 1 期の成果を

(注1)：方針９は当然の方針である。しかし、システムの設計に大きな影響を与えたので、あえて記載している。

踏まえ、第 2 期として、学部 3, 4 年生を対象とすることとなり [5,6]、北海道大学も参画することとなった。学内での種々の経緯や検討を経て、工学部共通科目「サイバーセキュリティ基礎演習」として、同分野の PBL 演習 A を開講することとなり、著者らが担当することとなった。

演習内容の検討に際し、学部共通科目とされたことから、必ずしも情報系の授業を履修していない学生の存在も想定しなければならないことや、情報系の学科・コースにおいても、直接関連する内容について、座学では一応の内容が提供されているものの実習形式では必ずしも十分に知識を会得できていないと考えられたことから「基礎演習」と名称を冠したこともあり、あまり特化した内容にせず、grep などを駆使した最低限の履歴の分析ができるよう、広範な背景知識を身につけられるよう、ともすれば平易な、ただし実際の機器操作を介してその内容を会得できるような内容とすることにした。

このため、各利用者に個別の UNIX 環境を用意し、演習において統一した環境で、UNIX 操作や正規表現の基本的用法を概観し、その上で、SSH（Secure SHell）の公開鍵作成と利用設定を、パスワード認証と対比させるべく、異なるプライベートネットワークを経由させて違いを実感させるような課題を与えた。また、TCP/IP 通信履歴の内容を適切に理解すべく、各々の UNIX 環境においてそれぞれ Web サーバプログラムを稼働させ、実際の通信の確立と履歴の蓄積を体感させた。それらの素地をもとに、グローバル IP アドレスを付して実際に稼働している著者のサーバで蓄積されている実際の履歴をそのまま渡し、表層的な類似に基づく分類とその解釈、また、いわゆる IP フィルタリングの結果を簡素に要約し高頻度のものを降順にグラフ表示する Python スクリプトを予め与え、履歴を扱いやすいフォーマットへ変換すること、グラフ表示に基づいて高頻度となった履歴に対する解釈を考えさせる課題を与え、最終的に班単位でのプレゼンテーションを行わせることを目標とした。なお、履修者の所属や知識、経験などを総合的に判断し、各班にはできるだけそれに即した課題を与えることとした。各班には与えられた課題に対するプレゼンテーション資料づくりを取り組ませ、演習の最後に全参加者の前でプレゼンテーションさせることとした。

令和元年度までは、夏期集中の形を取り、受講生を 2 日間、1 室に集め、授業資料をその場で手交し、数名単位でグループを構成、演習環境として参加者別にノート PC を与え、Windows10 内部の WSL（Windows Subsystem for Linux）を用いることで、当初環境を統一し、実施させてきた。しかしながら、コロナ禍に見舞われた令和 2 年度は、「1 室に集め」「授業資料をその場で手交」「参加者別にノート PC」がいずれも困難となることが、年度当初から容易に想定された。仮想クラウドサーバを履修者数購入する、仮想環境をダウンロードさせ、個々に保有する PC で稼働させる、などの方法も模索したが、履修希望者の半数以上が非情報系であることがわかり、いずれの形態も、利用に至るまでのところで困難が予想された。

以上の経過、条件から、利用者個々の計算機事情に依らず、統一的で、最終的に担当教員がスーパーバイズ可能な実習環境の実現を目指すこととした。

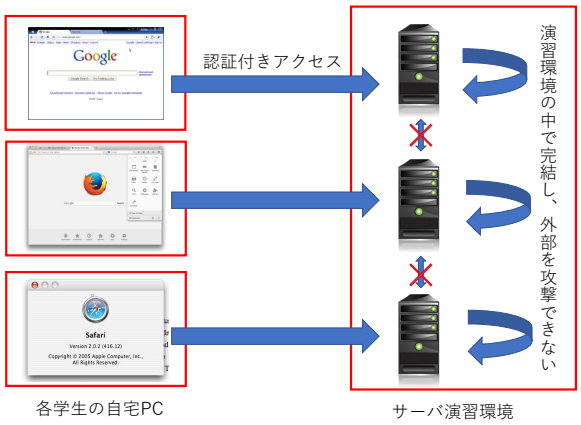



図 1　演習環境の概念図

## 3.　ブラウザ内で実現するオンライン演習環境の設計と実装

### 3.1　概　　要

本節では、ブラウザ内で実現するオンライン演習環境の設計と実装について述べる。1 節で述べたオンライン化に向けての 8 つの方針に基づき、学生の私物 PC の性能やソフトウェアの種類・バージョンに依存しにくい環境を提供するため、教員側で事前に準備するサーバ内に演習環境を構築し、学生はその環境に私物 PC 内にインストールされたウェブブラウザでアクセスすることとする (図 1)。

1 節で述べた通り、本演習はセキュリティに関する演習であり、セキュリティインシデント発生の抑止に特に気を配っている。そのため、学生がブラウザからサーバにアクセスする際は認証による本人確認を実施する（方針 8 [注2]）。さらに、方針 7 を実現するため、サーバ内での演習環境は学生ごとに分離する。

このような分離を実現するためには、従来 VMware, VirtualBox などの仮想マシンソフトウェアが使われてきた。しかし、本演習の履修者数は３０名を超えていて、教員、TA などの関係者の分を含めると、５０台程度の仮想マシンが必要となり、性能提供が困難になることが予想される。そこで、軽量な仮想マシンを提供するコンテナ技術を利用することとした [注3]。

### 3.2　設　　計

上述の設計要件を整理すると

- 既存の演習項目・内容を出来るだけ削らないこと
- セキュリティ上の要件
- 学生の環境・スキルに関する要件
- スケジュール上の制約

の 4 項目となる。以下これらの要件を満たすための設計を述べる。

多くのウェブサービスで使われている本人確認の方式として

(注2)：1 節で記載した「オンライン化の方針」の各方針のことを、以降では省略して「方針 8」のように記載する。
(注3)：著者らの同僚のクラウド技術の専門家からのアドバイスを参考とした。

（1） Web 上で電子メールアドレスを登録
（2） システムが登録者に対して電子メールを送信
（3） 利用者が上記電子メールのリンクをクリックすることで電子メールアドレスの正当性を確認する
があるが、これは著者（飯田）の経験上、実現に時間がかかり方針 9 を満たせないことがわかった。そこで、本人確認の方法として
（1） 学生に印刷した指導書を郵送する際に初期パスワードを記載したアカウント用紙を添付する（郵送には本人受け取り確認機能のついたレターパックプラスを利用）
（2） アカウント用紙の保持者のみがアカウント登録ができることとする
設計とした。

システムの設計・実装において、認証付きウェブの開発に時間がかかることが予想された。そこで、セキュリティ的にも安全が確認されていて、認証付きウェブの開発の効率が比較的高いと考えられる Laravel [7] を利用することとした。Laravel が提供する認証機能では、ID とパスワードが可能である。しかし、

- 郵送するアカウント用紙に記載した初期パスワード
- 初回ログイン時に本人が設定するパスワード（以下、本人パスワードと記載）

のどちらでもログイン可能とし、初期パスワード入力時に本人パスワードを再設定させる設計としたため、Laravel の認証機能を改造することとした。（このような設計にした理由は、学生が自分で設定したパスワードを忘れる可能性を考慮したためである。）

次に、演習で利用する UNIX 環境の提供は、ウェブブラウザ上で UNIX の CUI コマンドの実行ができる shellinabox [8] というソフトウェアを利用することとした。shellinabox は、JavaScript をつかって実装されており、xterm 互換のターミナル環境とコピーアンドペースト機能を提供している。ウェブブラウザ上で UNIX の CUI 環境を実現するソフトウェアは他にもあるが、コピーアンドペーストのテストに成功したソフトウェアはこれだけだったので、これを採用することとした。

shellinabox の環境は、方針 7 からユーザ毎に分離する必要がある。本システムでは、コンテナと呼ばれる仮想マシン単位で軽量に分離を実現する Docker [9] を用いて分離することとした。

次に、shllinabox のログイン画面を保護する方式を検討した。shellinabox 単独では、UNIX の login コマンドを介して、ID とパスワードによるログイン認証を行える。しかし、地理的にも契約している ISP の種類としても学生のアクセス場所は広域に分散しているため、サービスを広範囲に公開する必要がある。そのため、UNIX の ID とパスワードのログイン認証を生で公開するのは危険性が高いと判断し、Laravel によるウェブ認証でログインが完了している場合にのみ、ログインを許可する設計とした。

今回利用した Laravel のパッケージ Laradock(Docker を用いた Laravel 環境のパッケージ) [10] では、フロントエンドのウェブサーバとして nginx [11] を設け、その背後に php-fpm(注4) を設置する形式で実現されている。shellinabox を Laravel の認証の配下にいれるため、nginx をリバースプロキシ的に利用し、Laravel の認証状態を確認して shellinabox にフォーワードする構成をとることとした。

さらに、演習では ssh ログインの課題を実施するため、sshd の Docker コンテナを設けることとした。

以上の構成を図 2 にまとめる。Docker 環境内に設置する仮想ネットワークは 2 種類（laradock_frontend と laradock_backend）とする。これは、Laradock の標準構成である。図中の Docker 環境には、左から shellinabox (host01∼host74), sshd, php-fpm, mysql, phpmyadmin, nginx のコンテナなどを設置する。shellinabox に関しては、学生 50 名、教員 TA など 24 名の収容を目的として、host01 から host74 までのコンテナを設定する。sshd のコンテナは手動で固定 IP アドレスを割当、それ以外のコンテナは docker の動的 IP アドレス割当機能を利用する。また、host01∼host74 に関しては、Laravel Filemanager ソフトウェアを設定する。

sshd は ssh ログインの演習で利用するコンテナで、laradock_frontend からログインする場合はパスワード認証を許可することとし、laradock_backend の場合は公開鍵認証のみを許可することとする。これにより、従前の ssh 演習と同じ演習の実現を可能とする。

次の php-fpm, mysql, phpmyadmin, nginx は Laradock のコンテナである。php-fpm, mysql, nginx は Laradock の必須コンテナで、これらの組み合わせで Laravel の機能を実現する。phpmyadmin は mysql に登録するユーザ情報を管理するために利用する。

外部からの HTTPS(443/tcp) のアクセスは全て nginx コンテナが受付処理を行う。以下、nginx の設定概要を記載する。Larabel の PHP アクセスの場合は php-fpm コンテナの 9000/tcp にアクセスし、shellinabox の場合は host01∼host74 コンテナの 4200/tcp にアクセスする。さらに host01∼host74 コンテナは Laravel Filemanager 機能 [13] を利用するため、8080/tcp へのアクセスの設定を行う。host01∼host74 コンテナの 4200/tcp と 8080/tcp のアクセスは、Laravel によるウェブ認証完了後にのみアクセスを許可する設定とする。暗号化通信 TLS の終端処理は全て nginx コンテナで行うこととする。sshd コンテナは host01∼host74 コンテナとの ssh 通信を行うことのみが目的であるため、Docker 外へのサービス提供は行わない。

### 3.3 実　　装

本節では、前節で設計したブラウザを用いた演習システムの実装について述べる。表 1 にハードウェア環境を、表 2 にソフトウェア環境を示す。最大 20 人程度のユーザの同時利用が想定されており、Docker 環境の利用設計上、80 個程度のコンテナを同時に稼働する必要があることから、予算内で購入できる

(注4)：php-fpm は PHP の FastCGI 実装の一つで FastCGI Process Manager (FPM) [12] のことを指す。

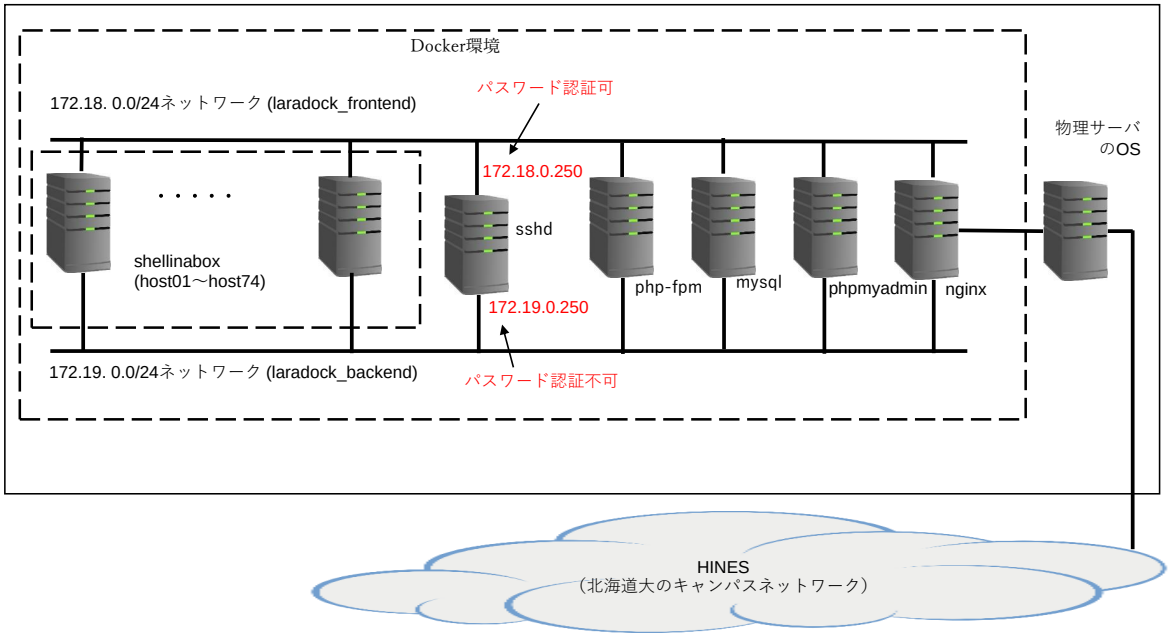



図 2　ネットワーク構成図

表 1　ハードウェア環境

| 名称 | 仕様等 |
|---|---|
| 機種名 | Dell PowerEdge R240 Server |
| CPU | Intel Xeon E-2276G 3.8GHz<br>(6 core, 12 thread) |
| メモリ | 32GB |
| ストレージ | 2TB HDD (7.2K RPM SATA) × 2<br>(ハードウェア RAID による RAID1 を設定) |

表 2　ソフトウェア環境

| 名称 | バージョン名など |
|---|---|
| OS | Ubuntu Server 20.04.1 LTS |
| SSL サーバ証明書 | UPKI 電子証明書発行サービス [14]<br>により取得 |
| Docker [9] | CE 版 (19.03.13) |
| Docker-Compose [15] | 1.25.0 |
| PHP | 7.3.21 |
| Laradock [10] | 2020/08/26 版 |
| Laravel Framework | 7.26.0 |
| nginx [11] | 1.19.2 |
| mysql-server | 8.0.21 |
| phpmyadmin | 5.0.2 |
| Portainer | 1.24.1 |
| shellinabox [8] | 2.21 |

範囲で性能のよいハードウェアを準備することとした。

設計に記載したソフトウェア群に設定・改造を施し、一部の Laravel と Python コードを新規作成することで演習環境を実装した。実装の際に困難だった箇所は

（1） Laravel の認証機能の改造及び追加実装

（2） 認証付き nginx の設定 (shellinabox)

（3） sshd コンテナへの固定 IP アドレスの付与 (Docker-Compose)

（4） Laravel Filemanager の導入・設定

等がある。ここではページ分量の都合で (1), (2) について要点を紹介する（3. 3. 1 節, 3. 3. 2 節）。なお、3. 3. 1 節, 3. 3. 2 節の説明は具体的な実装コードを紹介するため、興味の無い方はそれらの節を読み飛ばし 3. 3. 3 節から読むことをお勧めする。

### 3. 3. 1　Laravel の認証機能の改造及び追加実装

3. 2 節で述べた通り、初期パスワード及び本人パスワードのどちらを入力してもログインできるようにする。さらに、初期パスワードを入力した場合は本人パスワードの初期登録または更新が出来るようにする (注5)。そのために、mysql データベースに initialPassword というフィールドを追加し、Laravel の認証機能の実装である vendor/laravel/framework/src/Illuminate/Auth/GenericUser.php に getAuthInitialPassword() という関数を追加した。Laravel の認証設定 (config/auth.php) はドライバーとして'database' を採用した。

ポイントは、データベースによる認証の実装ファイル (vendor/laravel/framework/src/Illuminate/Auth/-DatabaseUserProvider.php) において、本人パスワードで認証に成功した場合は 1 を返し、初期パスワードの場合は 2 を返すように改造したことである。(7,8 行目) そのうえで、コントローラを実装する際に、認証結果が 1 の場合と 2 の場合で動作を変えるようにした。 (app/Http/Controllers/ExerciseController.php の 13~18 行目)

以下では、関係ある PHP ファイル群のうち、重要なものを抜粋して紹介する。その際、設定変更、新規開発、改造などの区別をカッコ内に示す。

`config/auth.php` (設定変更)

```
1  'providers' => [
2      'users' => [
```

(注5)：実装にあたっては [16, 17] を参考にした。

```
3           'driver' => 'database',
4           'table' => 'users',
5       ],
6   ],
```

routes/web.php (新規開発)

```
1   Route::get ('/exercise/auth',
2     'ExerciseController@getAuth')->name('auth');
3   Route::post('/exercise/auth',
4     'ExerciseController@postAuth');
```

vendor/laravel/framework/src/Illuminate/Auth/DatabaseUserProvider.php (改造)

```
1   public function validateCredentials(UserContract $user,
        array $credentials)
2   {
3       /* added by iida */
4       $result1 = $this->hasher->check( $credentials['
            password'], $user->getAuthPassword() );
5       $result2 = $this->hasher->check( $credentials['
            password'], $user->getAuthInitialPassword() );
6       //return $result1 or $result2;
7       if ($result2) return 2;
8       if ($result1) return 1;
9       return 0;
10  }
```

app/Http/Controllers/ExerciseController.php (新規開発)

```
1   public function getAuth(Request $request)
2   {
3       $param = ['message' => 'ログインしてください。'];
4       return view('exercise.auth', $param);
5   }
6
7   public function postAuth(Request $request)
8   {
9       $username = $request->username;
10      $password = $request->password;
11      $credentials = ['username' => $username, 'password'
            => $password];
12      $result = Auth::attempt($credentials);
13      if ($result == 1) {
14          return view('exercise.portal');
15      } else if ($result == 2) {
16          $item = DB::table('users')
17              ->where('username', $request->username
                    )->first();
18          return view('exercise.registration', ['form' =>
                $item]);
19      } else {
20          $msg = 'ログインに失敗しました。';
21          return view('exercise.auth', ['message' => $msg]);
22      }
23  }
```

resources/views/exercise/auth.blade.php (新規開発)

```
1   @extends('layouts.exercise')
2
3   @section('title', 'ユーザー認証')
4
5   @section('menubar')
6       @parent
7       ユーザー認証ページ
8   @endsection
9
10  @section('content')
11  <p>{{$message}}</p>
12      <form action="/exercise/auth" method="post">
13      <table>
14          @csrf
15          <tr><th>ユーザ名: </th><td><input type="
                text"
16              name="username"></td></tr>
17          <tr><th>パスワード </th><td><input type="
                password"
18              name="password"></td></tr>
19          <tr><th></th><td><input type="submit"
20              value="送信"></td></tr>
21      </table>
22      </form>
23  </p>
```

### 3.3.2 認証付き nginx の設定 (shellinabox)

次に、shellinabox を認証付きで実現するための nginx と Laravel の設定を述べる。shellinabox は 4200/tcp で動く HTTP サーバとして動作する。外部からは HTTPS でアクセスし、なおかつ Laravel での認証が成功した場合にだけアクセス出来るように設定する。

具体的には、Laravel 側にログインが完了したかどうかを確認するため、Laravel のコントローラに is_login という関数を設ける (ExerciseController.php の 1〜9 行目)。次に、nginx 側に、ログインが完了したかを確認する/_auth というロケーションを設ける (/etc/nginx/sites/default.conf の 1〜6 行目)。

実際の shellinabox の Docker コンテナ (host01〜host74) にアクセスするために、default.conf に/host/というロケーションを設ける (8〜19 行目)。/host/ロケーションは、/_auth ロケーションに対する認証確認を行う (default.conf の 9 行目)。また/host/ロケーションの設定は階層的になっていて、host01〜host74 の個別のロケーションの設定は/etc/nginx/ssl/host.conf で行う。各 host の URL は動的に変更されないため、個別の URL は推定されにくくする必要がある。そのため、乱数値を元に SHA224 でハッシュ値 (224bit=28Byte=ASCII56 文字) を作り、それを URL の一部とする。このハッシュ値は mysql のデータベースに url_hash というフィールドに保存しておく。url_hash の値は Laravel 側でアクセス出来るようにする (ExcerciseControler.php の 11〜16 行目)。実際に Laravel 上で shellinabox の URL をアクセスするページ (resources/views/excercise/-portal.blade.php) で url_hash の値を使って$url_1 を作る (9 行目)。$url_1 の値によるリンクを設ける (portal.blade.php の 15 行目)。

app/Http/Controllers/ExerciseController.php (Laravel)

```
1   public function is_login(Request $request)
2   {
3     if (Auth::check()) {
4       return response('OK', 200)
5         ->header('Content-Type', 'text/plain');
```

```
    } else {
      \App::abort(401, 'Not authenticated');
    }
  }
  中略
  public function url_hash()
  {
    $item = DB::table('users')
    ->where('id', Auth::user()->id)->first();
    return $item->url_hash;
  }
```

`/etc/nginx/sites/default.conf` (nginx)

```
  location /_auth {
    proxy_pass https://[サーバのFQDN]/exercise/is_login;
    proxy_pass_request_body off;
    proxy_set_header Content-Length "";
    proxy_set_header X-Original-URI $request_uri;
  }
  中略
  location /host/ {
    auth_request /_auth;
    proxy_http_version 1.1;

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Server $host;
    proxy_set_header X-Forwarded-For
          $proxy_add_x_forwarded_for;

    include /etc/nginx/ssl/host.conf
  }
```

`/etc/nginx/ssl/host.conf` (nginx)

```
location /host/乱数文字列 01 {
  proxy_pass http://host01:4200/;
}
location /host/乱数文字列 02 {
  proxy_pass http://host03:4200/;
}
後略
```

`resources/views/exercise/portal.blade.php` (Laravel)

```
@extends('layouts.exercise')
@section('title', 'Portal')
@section('menubar')
@parent
ポータルページ
@endsection
@inject('exp','App\Http\Controllers\ExerciseController')
@section('content')
<?php $url_1= "https://" . $exp->prefix() . "/host/" .
      $exp->url_hash(); ?>
<?php $url_2= "https://" . $exp->prefix() . "/webs/" .
      $exp->url_hash(); ?>
{{Auth::user()->name}}さん
<ul>
<li> <a href="/exercise/registration_new">個人情報確認・
      更新</a>
<li> <a href="/exercise/change-password">パスワード
      変更</a>
<li> <a href="{{$url_1}}" target="_blank" rel="
      noopener noreferrer">UNIX 演習環境ログイン</a>(
      ユーザ名: guest, パスワード: guest)
<li> <a href="/laravel-filemanager" target="_blank" rel
      ="noopener noreferrer">演習環境ファイル授受</a>
<li> <a href="{{$url_2}}" target="_blank" rel="
      noopener noreferrer">演習環境ウェブサーバ</a>
<li> <a href="/exercise/auth/logout">ログアウト</a>
</ul>
@endsection
```

#### 3.3.3 システムの動作概要

本節では、システムの動作概要を説明する。システムの画面例を図 3 に示す。トップページを開くと図 3(a) が表示され、ログインをクリックすると図 3(b) が表示される。ログインページにはユーザ名とパスワードの入力欄が表示される。初回ログイン時は郵送したアカウント用紙 (図 4) に記載されている、ユーザ名と初期パスワードを入力する。初回ログインの場合は履修者情報の初期登録ページ (図 3(c)) が表示される。

初期登録が終了する、あるいは、初期登録時に設定した本人パスワードでログインするとポータルページ (図 3(d)) が表示される。ポータルページでは、個人情報の更新、パスワード（本人パスワード）の変更ページへのリンクを表示すると共に、shellinabox のページ (図 3(e)) も表示できる。ポータルページにある通り、shellinabox のページのことは「UNIX 演習環境ログイン」と記載している。Laravel にログインしていない状態で、shellinabox の URL(静的に設定) を開くと、認証していないため表示できないことを示すページ (図 5) が表示される。

shellinabox のページでは、UNIX の login コマンドが起動するため、また別のユーザ名とパスワードの入力が要求される。すでに厳格な認証を行っているため、UNIX のユーザ名とパスワードは guest/guest で統一することとした。

shellinabox のページでは、UNIX の基礎演習 (リダイレクト, grep など)、ssh の演習 (図 3(f))、ネットワークの基礎演習 (dig, netstat, telnet によるサーバへの接続, apache サーバの起動など)、ログ解析演習 (grep および教員が事前に作った Python スクリプトを利用) を実施する。そのため、shellinabox の Docker コンテナを構成する際は、演習に必要なソフトウェア (Ubuntu パッケージ) を事前にインストールしている。

#### 3.3.4 管理者の管理手順

本節では、管理者の管理手順の概要を紹介する。

（1） Python スクリプトで乱数のパスワードに基づくアカウントファイル (CSV 形式) を作成

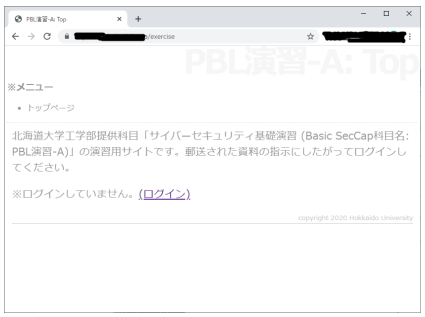
（2） CSV ファイルの内容を phpmyadmin 経由でデータベースに登録

（3） Word の差し込み印刷機能を使い、CSV ファイルからアカウント配布用紙 (図 4) を作成

以下に手順（1）の Python スクリプトの重要部分を示す。

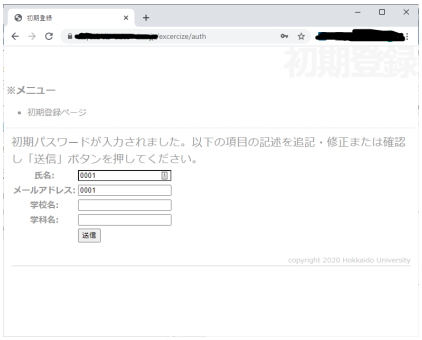
```
def pass_gen(size=12):
    return ''.join(secrets.choice(chars) for x in range(size))

for n in range(1,51):
    username='{:0>4}'.format(n)
```



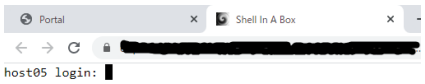
(a) トップページ

(b) ログインページ

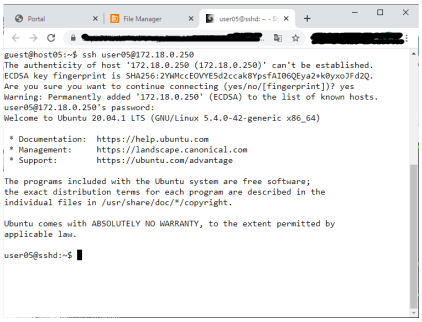


(c) 初期登録ページ

(d) ポータルページ



(e) shellinabox のページ (UNIX 演習環境)



(f) ssh の実行画面

図 3　システムの画面例

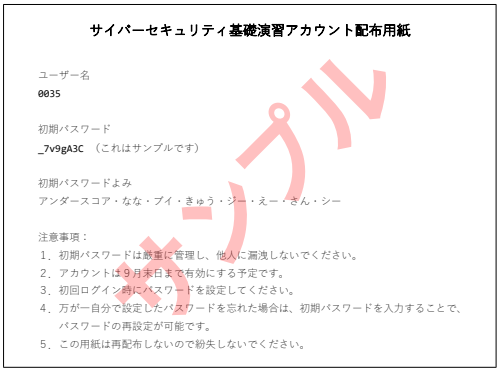



図 4　アカウント配布用紙

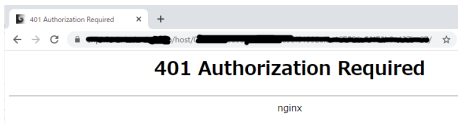



図 5　認証要求ページ

```
6   salt=bcrypt.gensalt(rounds=10,prefix=b'2b')
7   passwd=pass_gen(8)
8   utc_time=datetime.datetime.now(datetime.timezone.
        utc).strftime('%Y-%m-%d %H:%M:%S')
9   hashd=hashlib.sha224(pass_gen(32).encode('utf-8')).
        hexdigest();
10  print(username + ',' + passwd + ',' + bcrypt.hashpw(
        passwd.encode('utf-8'), salt).decode('ascii') +
        ',\"' + utc_time + '\",' + hashd);
```

上記スクリプトの CSV 形式の出力を説明する。第 1 カラムはユーザ名 (0001~0050) を、第 2 カラムは 8 文字の初期パスワードを、第 3 カラムはデータベースに記録する暗号化パスワードを、第 4 カラムは現在時刻を、第 5 カラムは 3.3.2 節の url_hash の値を示す。第 2 カラムの初期パスワードはアカウント配布用紙に利用する。それ以外のカラムの値は phpmyadmin を経由して mysql のデータベースに登録する。

## 4.　演習の実施

本節では、前節で紹介した演習システムを利用して実際に実施した演習の状況を紹介する。

- 北海道大学科目名：サイバーセキュリティ基礎演習 (工学部共通科目)
- enPiT-Security 科目名：PBL 演習-A
- 初回実施グループ

  日程：2020 年 9 月 18 日（金)~9 月 19 日（土）
  履修学生数: 16 名

- 第 2 回実施グループ

  日程：2020 年 9 月 24 日（木）~9 月 25 日（金）
  履修学生数：15 名

- 履修学生
  - 北海道大学・工学部所属学生：18 名 (うち情報系 5 名)
  - 北海道大学・他学部所属学生：1 名
  - 参加校など (2 大学 1 高専)・所属学生：12 名
- 実施時間
  - 13:00~18:00 (全 4 日間共通)

学生を初回と第 2 回実施グループに分けたのは、遠隔演習でも従前の対面演習と出来るだけ近づけるため、1 回あたりの受講者数を 20 名以下としたためである。演習では Zoom を用い、講師が各課題の説明をした後に、Zoom のブレークアウトセッションの機能を使って、班に分かれて班ごとにグループワーク

を行ってもらった(注6)。各班内のコミュニケーションは Zoom による音声通話の他に、Slack による文字やファイル交換の環境を提供した。

班ごとに分かれる際に対応するため、2 名の教員（南、飯田）の他に大学院生の TA 1 名を配置した。さらに参加校の教員や連携企業の方にも TA 役を担ってもらい、各班にできるだけ 1 名の指導者が常時配置できる体制を構築した。

以下に、演習の実施状況を紹介する。まず、事前にテストをしていたが、大人数でアクセスすると負荷が高く、演習に支障がでるかもしれないことを懸念していたが、実際は負荷の問題はまったく生じなかった。また、想定外のトラブルが起こり、演習の進行に支障がでることを懸念していたが、大きなトラブルはあまり起こらず、演習は滞りなく進んだ。

小さなトラブルとしては shellinabox のログイン画面 (図 3(e)) で、「パスワードが入力できません」という学生が何名か発生した。それは、普段使っている Windows などの OS では、パスワードを 1 文字入力する度に●が表示されることになれていて、UNIX の CUI ログイン画面で文字を入力したのに●が表示されないので入力ができていないと考えたようであった。初回グループでそのような質問が発生したので、第 2 回グループでは、説明資料を追加して対応した。

また、Laravel Filemanager の設定が不完全であり、演習環境とのファイルの授受の方法がわからない学生がいたので、それについても説明資料を追加して、演習環境からファイルをダウンロードする方法の手順を説明した。

次に、Zoom のブレークアウトセッションの使い方について得た知見を紹介する。初回実施グループでは、班の数と同数のブレークアウトセッションを用意していた。しかし、経験値が低い学生が班内の他の学生から大幅に遅れるケースがあり、その場合、教員とその学生だけの特別なブレークアウトセッションがあると対応がしやすいことがわかった。そのため、第 2 回実施グループの演習の際は、班の数よりも多めのブレークアウトセッションを準備することで、そのような学生の対応を行うこととし、実際に有効であった。

演習時は、連携企業の方および enPiT-Security 事業のアドバイザー委員などに見学をいただいた。見学者からは、演習の内容そして演習の環境について、概ね好意的な評価をいただいた。

履修した学生からは

- Windows からでも、特別なソフト等をインストールしなくても UNIX 環境が快適に使えて良かった
- 専門分野外の科目だったので新鮮だった
- UNIX の実践力が身に付いた
- 本当は対面形式で履修したかった
- 演習内容の分量が多く、グループプレゼンテーションの準備時間が不足していた

などの感想が寄せられた。次に、上記の最後の感想を補足する。「オンラインでも従来とできるだけ同じ演習を提供する」ことを目標としたため、詰め込みすぎになっていたかもしれない。今後オンライン形式で同様の演習を実施する際は、演習の分量などを調整する必要があると考えられる。

(注6)：初回実施グループ、第 2 回実施グループ共に 4 班に分かれてもらった。

## 5. おわりに

本稿では完全オンライン形式で演習を実施するために構築した、演習システムについて紹介し、そのシステムを利用して実施した演習の状況等を説明した。


## 謝　辞

仮想化ソフトウェアの利用などについてアドバイスをいただいた、北海道大学情報基盤センター長の棟朝雅晴先生に感謝いたします。また、演習の運営に協力いただいた、参加校の教員のみなさま、連携企業のみなさま、TA の渡邊君に感謝します。演習のオンライン化に関し各大学の準備状況などの情報提供をいただいた、enPiT-Security 運営委員会ご関係の皆様に感謝いたします。なにより、コロナ禍の中でオンライン演習に参加してくれた履修学生の皆様に感謝します。


## 文　献


[1] 北海道大学・オープンエデュケーションセンター, “北海道大学におけるオンライン授業導入ガイド,” https://sites.google.com/huoec.jp/onlinelecture, 2020 年 10 月 16 日アクセス.

[2] 国立情報学研究所, “4 月からの大学等遠隔授業に関する取組状況共有サイバーシンポジウム,” https://www.nii.ac.jp/event/other/decs/, 2020 年 10 月 16 日アクセス.

[3] “成長分野を支える情報技術人材の育成拠点の形成 (enPiT): 高度 IT 人材を育成する産学協働の実践教育ネットワーク,” http://www.enpit.jp/, 2020 年 10 月 16 日アクセス.

[4] “実践的なセキュリティ人材を育成する (enPiT-Security),” https://www.seccap.jp/basic/, 2020 年 10 月 16 日アクセス.

[5] Y. Kanaya, D. Kotani, K. Iida, and H. Sone, “Ethical Education on Information Security Mind for Practical Security Learning,” *IEICE Tech. Rep.,* vol. 120, no. 177, IA2020-2, pp. 6–9, Oct. 2020.

[6] 和泉諭, 庄子栄光, 水木敬明, 菅沼拓夫, 中尾光之, 曽根秀昭, “東北大学における enPiT 第 2 期セキュリティ分野 Basic SecCap の取り組み利用統計を見る,” コンピュータセキュリティシンポジウム *2017* 論文集, no. 2, pp. 1449–1452, 2017 年 10 月.

[7] “Laravel: The PHP Framework for Web Artisans,” https://laravel.com/, 2020 年 10 月 16 日アクセス.

[8] “shellinabox”, https://github.com/shellinabox/shellinabox, 2020 年 10 月 16 日アクセス.

[9] “Docker,” https://docker.com, 2020 年 10 月 16 日アクセス.

[10] “Laradock”, https://laradock.io/, 2020 年 10 月 16 日アクセス.

[11] “nginx”, https://nginx.org/en/, 2020 年 10 月 16 日アクセス.

[12] The PHP group, “FastCGI Process Manager,” https://www.php.net/manual/en/book.fpm.php, 2020 年 10 月 16 日アクセス.

[13] UniSharp, “Laravel Filemanager,” https://github.com/UniSharp/laravel-filemanager, 2020 年 10 月 16 日アクセス.

[14] 国立情報学研究所, “UPKI 電子証明書発行サービス,” https://certs.nii.ac.jp/, 2020 年 10 月 17 日アクセス.

[15] “Overview of Docker Compose,” https://docs.docker.com/compose/, 2020 年 10 月 16 日アクセス.

[16] 竹沢有貴, 栗生和明, 新原雅司, 大村創太郎, *PHP* フレームワーク *Laravel: Web* アプリケーション開発, ソシム, 2018 年 10 月.

[17] 掌田津耶乃, *PHP* フレームワーク *Larabel* 入門*:* 第 *2* 版, 秀和システム, 2020 年 1 月.