Vol.2021-CLE-33 No.11
2021/3/25

ネットワークセキュリティ演習のための直感的でシームレスな操作と軽快な応答性を目指したウェブ型演習システムの開発

立岩 佑一郎[1,a)]

概要：筆者の担当するネットワークセキュリティ演習では，受講者は仮想マシン User-mode Linux（UML）を機器としたネットワークを構築し，そのネットワークにて攻撃や防衛のツールを実行する．従来の演習システムでは，受講者は windows アプリケーションにより UML を稼働させる遠隔サーバのデスクトップ画面にアクセスし，コマンド実行により UML を設定することでネットワークを構築していた．しかしながら，コロナウィルス感染対策などから自宅での演習を考えた場合，特定の OS に依存したアプリケーション，細いネットワーク回線，および UML の設定ミスへのサポートの難しさなどから，受講者が効率的に演習を行えないことが予想される．本稿では，直感的でシームレスな操作と軽快な応答性を目指したウェブ型演習システムを提案する．この演習システムは，機器の配置やケーブルの接続などをマウス操作で直感的に編集できるウェブページ，UML のコンソールに接続されたウェブページ，UML で実行された X クライアントを操作できるウェブページを提供する．

1. はじめに

User-mode Linux[1]（以降，UML と呼ぶ）は Linux 上で動作するプロセスであり，Linux 計算機を仮想的に実現する．また，UML の付属ツールである uml_switch はスイッチングハブのように動作する．uml_swtich に UML のネットワークインタフェースを接続することで，仮想的なネットワーク（以降，仮想 NW と呼ぶ）をそのホスト計算機上に実現できる．

これまで，筆者の担当するネットワークセキュリティ演習では，受講者は演習室の Windows 端末にて vSphere Client[2] を起動し，それを通じて遠隔サーバ上で稼働する Debian Linux 3.1 のデスクトップ画面を操作した．**図 1** は，クライアントとサーバをスイッチングハブで接続した仮想 NW である．クライアントやサーバのウィンドウは，各々のコンソールとして動作している．これらの仮想機器の生成や仮想機器間の接続は Debian Linux のコンソールにて UML のコマンドを実行することで実現される．受講者は，機器の配置・撤去，機器間の接続・切断，および機器での操作（例えば，攻撃や防衛ツールの実行）により，脆弱性のあるネットワークにて攻撃を成功させたり，そのネットワークを防衛できるように変更したりする．

しかし，この演習環境には以下のような問題点がある．

(1) コロナウィルス感染対策のため受講者を自宅で演習させたいが，以下の原因により受講者が自宅に演習環境を構築するのが難しい．
- OS やスペックにばらつきのある PC やタブレットのため，windows アプリケーションである vSphereClient が動作しないものがある．
- 回線品質にばらつきのあるネットワーク環境のため，画面転送によるデスクトップ画面の操作では快適に作業できないことがある．
- 受講者とのやりとりが電話やメールとなるため，教師や TA が演習環境の構築をサポートしづらい．

(2) 以下の原因により，受講者が効率的に演習に取り組めない．
- コンソールの区別が付きにくいため，目的の UML のコンソールではなく，ホストのコンソールや別の UML のコンソールを誤って操作してしまう．

[1] 名古屋工業大学
NIT, Gokiso-cho, Showa-ku, Nagoya-shi, Aichi 466–8555, Japan
[a)] tateiwa@nitech.ac.jp

- 仮想 NW のネットワークトポロジーはコマンド実行により形成される．このとき，コマンド引数の設計に手間取ったり，コマンド入力を間違えてしまい，目的の仮想 NW を早く正確に作成できない．
- ネットワークトポロジーを正しく思い出せずに，目的と異なる作業をしてしまう．

(3) セキュリティホールは発見されるたびに埋められているため，演習したい攻撃によっては古い OS やツールが必要となる．このため，様々なバージョンの UML からなる仮想 NW を構築したいが，以下の原因により難しい．

- UML を稼働させる OS（以降，ホスト OS と呼ぶ）は UML のバージョンに依存するため，様々な UML を混在させた仮想 NW を単一のホスト OS では実現困難である．
- 異なるホスト計算機で稼働する UML や uml_switch を直接的に接続する機能をそれら自身が有していない．

本研究では，UML を用いたネットワークセキュリティ演習のための，直感的でシームレスな操作と軽快な応答性を目指したウェブ型演習システムを提案する．このシステムは下記の特徴を持つ．

- 遠隔サーバ上に仮想 NW を実現し，それを操作できる動的なウェブページを提供する．また，UI とサーバ間の通信に文字転送と画面転送を併用する．これらにより，受講者はブラウザを通じて演習環境を快適かつ容易に利用できる（問題点 1 を解決）．
- 仮想 NW のトポロジーをマウスで操作できる動的なウェブページを提供する．さらに，そこに描画されたネットワークに UML のコンソールが対応づけられている．これらにより，受講者はトポロジーを直感的に操作して仮想 NW を形成できる（問題点 2 を解決）．
- 様々な遠隔サーバ上にて稼働させた仮想機器間でイーサネットフレームを転送できるトンネルで接続したオーバレイネットワークを形成する．これにより，様々なバージョンの UML を含む仮想 NW を形成できる（問題点 3 を解決）．

2. 関連研究

NetPowerLab[3] は，遠隔サーバ上の UML を仮想機器とする仮想 NW をユーザに提供する．ユーザは，ウェブ UI によりネットワークトポロジーをマウス操作で直感的に編集したり，仮想機器のコンソールを利用したりできる．しかし，仮想機器の X クライアントを利用できなかったり，様々なバージョンの UML から構成されるネットワークを利用できなかったりするため，本演習には向かない．

V-Lab[4]，CloudLab[5]，および ThoTh Lab[6] は，遠隔サーバ上の仮想 NW をユーザに提供する．仮想機器として，V-Lab で Xen，CloudLab で Xen と Docker，ThoTh Lab で QEmu が用いられている．まず，ユーザはウェブページでネットワークトポロジーや仮想機器の構成を定義し，それをシステムに登録する．その後，その定義に基づいてシステムが仮想 NW を生成する．最後に，ユーザは仮想 NW を選択し，V-Lab と ThoTh Lab ではウェブページで仮想機器での操作を行い，CloudLab ではユーザの用意・設定した SSH クライアントや X サーバにより仮想機器での操作を行う．しかし，ネットワークトポロジーや仮想機器の構成を定義するフェーズ，その定義に基づいてネットワークを生成するフェーズ，各機器を操作するフェーズに分かれているため，トポロジーの変更と機器での操作を効率的に繰り返せない．このため，本演習には向かない．

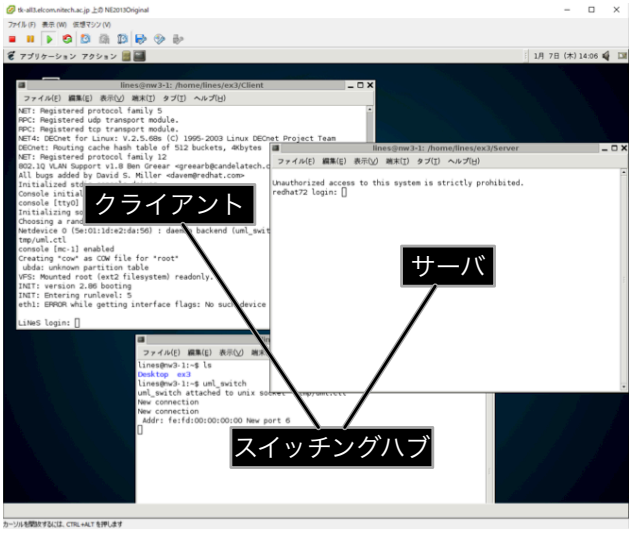

図 1　旧演習システム
Fig. 1　Previous version of the exercise system.

3. 基盤技術

3.1 User-mode Linux

UML は Linux 上で動作するプロセスであり，Linux 計算機を仮想的に実現する．UML のカーネルプログラム（Linux のバイナリ）にイメージファイル（ディストリビューションのファイル）を引数として実行すると，UML のプロセスが稼働される．UML のホスト OS で管理者権限を持っていないユーザであっても，UML 内で管理者権限を持つことができ，ソフトウェアをインストールしたり，サービスを提供したりできる．また，オリジナルイメージからの変更分を差分ファイルで保存していくため，イメージのコピーなしに（すなわち，即座に）新しい機器を用意できる．

しかし，新しいホスト OS で古い UML のカーネルプログラムを実行できなかったり，その逆もあったりする．すなわち，新旧の UML を一つのホスト OS 上で混在して稼働させることが難しいと言える．セキュリティホールは見つけられたり塞がれたりを繰り返し，クラッキングツール

は進化し続けている．このため，新旧の UML が混在する仮想 NW が演習に望まれる．

3.2 TAP と Bridge

Linux の TAP は仮想的なネットワークインタフェースであり，イーサネットフレームを取り扱える．UML 内部のネットワークインタフェース（例えば，eth0）とホスト OS 上に生成した TAP（例えば，tap0）を接続すると，UML 内部のアプリケーションはイーサネットフレームを TAP を通じて UML 外部と送受信できる．

Linux の Bridge は仮想的なネットワークインタフェースであり，ネットワークスイッチのようにイーサネットフレームを転送する．一つの Bridge に複数の TAP を接続することができる．UML に接続された TAP を Bridge に接続することで，それらの UML をネットワークスイッチで接続したような仮想 NW を形成できる．これらの UML のネットワーク設定は，ホスト OS やその他の UML のネットワーク設定に干渉しない．

3.3 トンネリング

トンネリング技術の中には，トンネルで接続された機器間でイーサネットフレームの転送を可能にするものがある（以降，イーサネットトンネルと呼ぶ）．ある二つの機器間にトンネルを作成すると，これらの機器にはトンネルの出入り口となる仮想的なネットワークインタフェース（以降，トンネル口と呼ぶ）が作成され，このトンネル口を通じてイーサネットフレームが交換される．これにより，あるホストの TAP や Bridge とこのトンネル口を接続することで，別のホストのトンネル口に接続された TAP や Bridge にイーサネットフレームを転送できるため，異なるホストの UML や Bridge の間でイーサネットフレームを交換できる．

4. 演習概要

演習では**表 1** の達成目標に対して，受講者は指導書に掲載されている課題に取り組む．攻撃や防衛を行うための実験環境としてのネットワークを構築し，そのネットワークにて各種ツールを実行し，その経過や効果を観察する．その後，ネットワーク管理者やクラッカーなどが登場し，攻撃成功や防衛成功などが起きる物語を考案の上，その攻撃や防衛などをネットワークで実験する．

5. 実現方法

5.1 システム概要

仮想機器であるサーバ，クライアント，ルータ，ファイアウォールを UML で，スイッチングハブを Bridge で実現する．

図 2 にシステム構成を示す．模擬サーバは設定サーバからの遠隔操作を受け付け，仮想機器を稼働させて仮想 NW を形成する．設定サーバは，仮想機器間の接続関係を編集するためのウェブページ（以降，トポロジーページと呼ぶ）と，UML のコンソールとの入出力のためのウェブページ（以降，コンソールページと呼ぶ），および UML の X クライアントとの入出力のためのウェブページ（以降，X ページと呼ぶ）を提供する．中継サーバは，自身と模擬サーバとの間に作成されたイーサネットトンネルのトンネル口を Bridge で接続することで，二つの模擬サーバのトンネル口間でイーサネットフレームを交換できるようにする．端末サーバはコンソールページと UML のコンソールとの入出力を中継する．X 転送サーバは，X サーバとして UML の X クライアントからの描画要求を受け付け，その出力を X ページに描画する．また，X ページからのユーザ入力を受け付け，X クライアントに送信する．Windows，Mac，Android や IOS のウェブブラウザを通じて，受講者は仮想 NW を管理する．

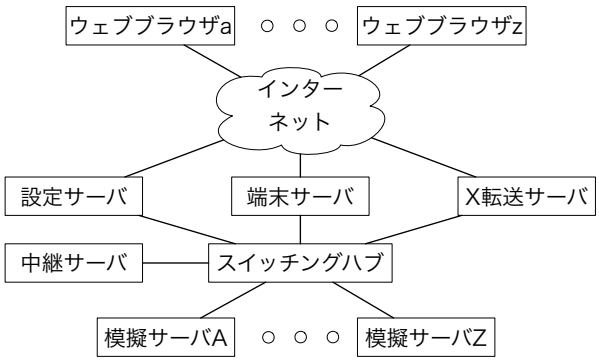

図 2 システム構成
Fig. 2 System overview.

UML のホスト OS は UML のバージョンに依存し，演習では古いバージョンの UML を用いたいことがある．一方，新しい OS の方が安全かつ効率的にウェブ UI を開発・運用できるので，そのためのサーバには新しい OS が望ましい．このため，UML のホスト計算機（模擬サーバ）とその他のサーバ計算機とを分け，模擬サーバをインターネットからアクセス不可能にした．

5.2 トポロジーページ

トポロジーページはネットワークトポロジー，仮想機器の一部設定と状態，それらに対するユーザ入力を受け付ける．応答性を高めるため，一部のデータをキャッシュし，そのキャッシュに基づいてユーザの一部の入力に対して応答を発行する（すなわち，設定サーバで処理しない）．例えば，接続できない機器の組み合わせの検証や仮想機器アイコンの移動による描画データの計算である（一定間隔でサーバに反映される）．

html5 の svg 要素によりネットワークトポロジーを描画する．Document Object Model（DOM）により，ネット

表 1　ネットワークセキュリティ演習の概要
Table 1　Summary of the network security exercise

	作業内容	達成目標
ネットワークの構築	• サーバ，クライアント，スイッチングハブ，ルータ，およびファイアウォールでネットワークを構築する	• サーバソフトウェアの基本的な設定を行える • アプリケーションプロトコルの概念を説明できる • 通常使用時のシステムのログを解釈できる
攻撃	• ツール実行によりネットワークを攻撃（DoS/DDoS，パスワードクラッキング，パケット盗聴，バックドア，バッファオーバフロー，セッションハイジャック，ルートハッキング）する • 攻撃中のシステムとネットワークの反応およびログを観測する	• 各種攻撃の仕組みについて説明できる • システム＆ネットワークの反応およびログから攻撃を感知できる
防衛	• iptables によりパケットをフィルタリングする	• FW の仕組みについて説明できる • 要求に対して適切なルールセットを設計できる • iptables の設定を行える
クラッカープログラミング	• セキュリティホールのあるプログラムを作成する • クラッキング用のプログラムを作成する • セキュリティホールに対するパッチを作成する	• 安全なプログラムを設計できる • プログラム中の危険箇所を見つけられる • プログラム中の危険箇所を修正できる

ワークトポロジーの描画をマウス操作で編集可能にする．具体的には，機器の配置，撤去，移動，起動，停止および機器間のケーブル接続，切断をアイコンのマウス操作により行う．

XMLHttpRequest により，ページ遷移をすることなく，設定サーバに要求/応答の通信を非同期で行う．編集内容を要求として設定サーバに送り出し，設定サーバの処理結果を応答として受け取る．例えば，受講者がトポロジーページで機器を移動させたとする．このとき，ブラウザは設定サーバにその要求を送り出し，それに対する応答を待たずに，受講者からの入力を受け付ける．これにより，受講者は機器の移動結果に影響されない編集に取りかかることができる．

5.3　設定サーバ

設定サーバはトポロジー管理ページからの要求を受け，**表 2** に示すように処理をした後に，応答を返す．表中の台帳は自身が管理し，中継サーバや模擬サーバでの操作は遠隔実行している．

稼働させられる UML のカーネルバージョンは模擬サーバの OS に依存する．また，サーバ計算機のリソース上限から，一つの計算機ですべてのユーザのプロセスを同時に稼働させることはできない．このため，設定サーバは，機器を実現するプロセスをその機器の種類とユーザに応じた模擬サーバにて稼働させる．同じ模擬サーバで稼働する UML と Bridge による仮想 NW は，UML の TAP と Bridge を接続することで形成される．異なるサーバで稼働する UML と Bridge による仮想 NW は，UML の TAP と Bridge をイーサネットトンネルで接続することで形成される．

表 2　模擬サーバでの要求処理
Table 2　Request processing in simulation servers.

要求	手続き
UML 配置	台帳に登録し，一意になる TAP 名を生成し，模擬サーバにて TAP を生成する
スイッチ配置	台帳に登録し，一意になる Bridge 名を生成し，模擬サーバにて TAP を生成する
ケーブル接続	台帳に登録し，機器を稼働させる模擬サーバに応じて接続処理を進める．模擬サーバが同一であれば，Bridge に UML の TAP を接続する．そうでなければ，中継サーバに各模擬サーバからイーサネットトンネルを構築し，各模擬サーバにてトンネル口と UML の TAP を Bridge で接続またはトンネル口を Bridge に追加し，中継サーバのトンネル口を Bridge で接続する．
UML 撤去	模擬サーバにて TAP を削除し，台帳から削除する
スイッチ撤去	模擬サーバにて Bridge を削除し，台帳から削除する
ケーブル切断	機器を稼働させる模擬サーバが同一であれば，Bridge から UML の TAP を削除する．そうでなければ，各模擬サーバにてトンネル口と UML の TAP を Bridge から削除または Bridge からトンネル口を削除し，中継サーバのトンネル口を Bridge から削除し，最後に，台帳から削除する．
機器移動	台帳を更新する
機器起動	UML を起動するコマンドまたは Bridge を UP 状態にするコマンドを模擬サーバにて実行し，台帳を更新する．
機器停止	UML を停止するコマンドまたは Bridge を DOWN 状態にするコマンドを模擬サーバにて実行し，台帳を更新する．

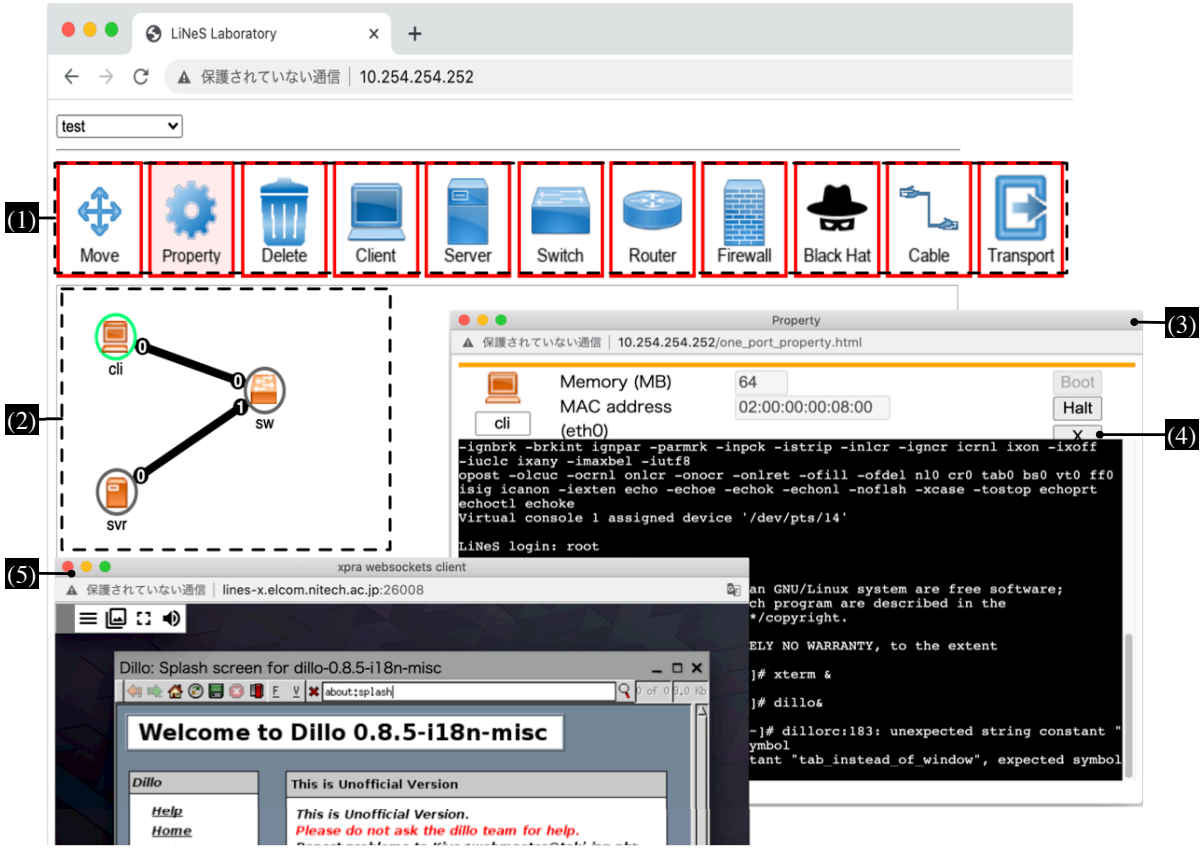

図 3 ネットワークを管理するためのウェブページの例
Fig. 3 Example of web pages for managing networks.

6. プロトタイプ

設定サーバを node.js[7]，トポロジーページを javascript，jquery[11]，ajax，SVG.js[9]，コンソールページを Xterm.js[12]，端末サーバを socket.io[10]，模擬サーバの遠隔コマンド受付を openssh[8]，イーサネットトンネルを vtun[13]，X 転送サーバおよび X ページを Xvfb および xpra にて実装した．

図 3 は Mac 版の Google Chrome での実行例で，上からトポロジーページ，コンソールページ，X ページが表示されている．図中 (1) から機器を選択した状態で，領域 (2) をクリックすると，選択した機器が描画される．ウィンドウ (3) は機器 cli のプロパティ画面であり，ウィンドウ下部のコンソール領域は模擬サーバ上で稼働している UML のコンソールを描画し，キーボード入力を受け付ける．ボタン (4) を押すと，機器 cli の X ウィンドウマネージャを表示するウィンドウ (5) が表示される．ウィンドウ (5) では，機器 cli で起動したブラウザ（X クライアント）が表示されている．なお，領域 (2) の機器を表すアイコンはマウスドラッグにより移動できる．

7. おわりに

遠隔サーバ上で稼働する UML によるネットワークをネットワークセキュリティ演習の受講者が効率的に利用するための演習環境を提案した．同一サーバ上であれば UML 間を Linux Bridge により接続し，異なるサーバ上であれば UML 間を Linux Bridge とイーサネットトンネルにより接続することで，仮想 NW を実現する．受講者はウェブブラウザに提示された，仮想機器間の接続関係を編集するためのウェブページと，UML のコンソールとの入出力のためのウェブページ，および UML の X クライアントとの入出力のためのウェブページで，ネットワークを管理する．

今後の課題として，操作性と応答性の評価が挙げられる．前者のためには，ネットワークの品質を無視できるよう高速ネットワークにおいて，演習に必要な作業を直感的でシームレスに行えたかを被験者のアンケートで評価する．後者のためには，様々な遅延や帯域のネットワークにおいて，演習に必要な作業を快適に行えたかを被験者のアンケートで評価する．

謝辞 本研究は JSPS 科研費 20K12108 の助成を受けたものです．

参考文献

[1] The User-mode Linux Kernel Home Page, 入手先 ⟨http://user-mode-linux.sourceforge.net/⟩(2021.02.22).

[2] vSphere Client の使用, 入手先 ⟨https://docs.vmware.com/jp/VMware-vSphere/5.5/com.vmware.vsphere.hostclient.doc/GUID-DAB486D6-3E33-4939-B80A-BB17CB3B4E1E.html⟩(2021.02.22).
[3] Nobukazu Iguchi: *Development of a self-study and testing function for NetPowerLab, an IP networking practice system*, International Journal of Space-Based and Situated Computing, Vol. 4, No. 3/4, pp.175-183, Nov. 2014.
[4] Le Xu, Dijiang Huang, Wei-Tek Tsai: *Cloud-Based Virtual Laboratory for Network Security Education*, IEEE Transactions on Education, Vol. 57, No. 3, pp.145-150, Aug. 2014.
[5] CloudLab, 入手先 ⟨https://www.cloudlab.us/⟩(2021.02.22).
[6] ThoTh Lab, 入手先 ⟨https://www.thothlab.com/⟩(2021.01.28).
[7] Node.js, 入手先 ⟨https://nodejs.org/en/⟩(2021.01.28).
[8] OpenSSH, 入手先 ⟨https://www.openssh.com/⟩(2021.01.28).
[9] SVG.js, 入手先 ⟨https://svgjs.com/docs/3.0/⟩(2021.01.28).
[10] Socket.IO, 入手先 ⟨https://socket.io/⟩(2021.01.28).
[11] jQuery, 入手先 ⟨https://jquery.com/⟩(2021.01.28).
[12] Xterm.js, 入手先 ⟨https://xtermjs.org/⟩(2021.01.28).
[13] VTun - Virtual Tonnels over TCP/IP networks, 入手先 ⟨http://vtun.sourceforge.net/⟩(2021.01.28).