IaaS を活用した
情報セキュリティ演習環境の設計と実装

小谷 大祐 (京都大学)

http://bit.ly/itrc45-basic-seccap-kyoto

「情報セキュリティ演習」

- Basic SecCap (enPiT-Security) で京都大学が開講する演習科目の一つ (通称 PBL演習M)
  - 京都大学工学部の正式科目、夏期集中 (9月)

※Basic SecCap 7：専門科目2単位、演習科目1単位、および基礎科目4単位の合計7単位以上を取得
※Basic SecCap 8：Basic SecCap 7授与要件に加え、先進演習科目より1単位を取得
※Basic SecCap 10：Basic SecCap 7授与要件に加え、先進演習科目より大学院インターンシップを含む計3単位を取得

「情報セキュリティ演習」の内容

外部からの不正アクセスの試みを検知する侵入検知システム（IDS）では、膨大な数の警報が発せられ、その解析は人手では困難である。ここでは、**IDSの仕組みと役割を学んだ上で、機械学習によりIDSの警報ログから正常通信と攻撃を分類する演習**を実施する。

- 攻撃を見つけるというのはどういうことなのかを学ぶ
  - ネットワーク型侵入検知システム（NIDS）の仕組みと役割
- 攻撃とはどのようなものかを理解する (攻撃してみる)
- 一般的なシグネチャ型 NIDS (Snort) がどのように攻撃を検知しているのかを実践を通して理解する
  - 自分でしてみた攻撃を検知するシグネチャを書く
  - False Positive や False Negative があることを体験する
- 機械学習を用いてより攻撃の通信の検知を試みる
  - Kyoto2006+ 準拠のトラフィックデータから正常通信と攻撃を分類

演習に必要な演習環境

- 攻撃・IDS による攻撃検知を実践できる
  - NIDS がリアルタイムに通信を監視するには root 権限が必要
  - 攻撃するホスト/されるホスト/IDS の最低3台必要
  - 誤って外に攻撃してしまうことがないように
- 機械学習の実践によく利用されるソフトウェアが利用可能
  - Jupyter Notebook + 多くのライブラリ (scikit-learn など)
  - 経験のある学生が自分で使いたいライブラリを追加できる
  - 演習中に一からインストールする時間はない
- 数GB〜数十GBのトラフィック (セッション) データを扱える

演習環境を設計する上での考慮

- Basic SecCap の枠組みで他大学の学生を受け入れるが受講生数が読みにくい (必要な機材が必要かわからない)
  - Basic SecCap 参加校数・受講生数とも毎年増加
  - 本学の演習を履修するかどうかは受講生の希望次第
- enPiT への補助金が終了しても継続できる (人・お金)
  - 正式科目なのですぐにはやめられない
- 2日間だけ動けばよい
  - 通年で計算資源を持ち続ける必要はない
- 担当者 (=私) の負担を減らす
  - つまらない単純作業に時間を費やすのは避けたい

方針

受講生に root を渡せる、
誤って外部に攻撃することを防げる、
複数台のホストが必要。

→ 隔離したネットワークに接続した
VMを構築し、受講生に提供。

機械学習によく利用される
ソフトウェアをあらかじめ準備。
希望により追加インストール可。

→ 上記のVMに予めインストール。
追加時は教員に申し出て
一時的にインターネットに接続。

受講生数が読みづらく、
必要な機材の量が予想しにくい。
数GB〜数十GBのトラフィック
データを扱える。

→ 受講者数・データサイズに応じた
VM を IaaS 上に毎年構築。
VM にアクセスするための端末は
受講生に持ち込ませる (BYOD)。
今回はさくらのクラウド

方針 (その他)

enPiT の補助金が終了しても継続できる (人・お金)。

→ 補助金が終わったら別途措置されるはずの予算額に応じて定員を設定。

2日間だけ動けばよい。
担当者の負担を減らす。

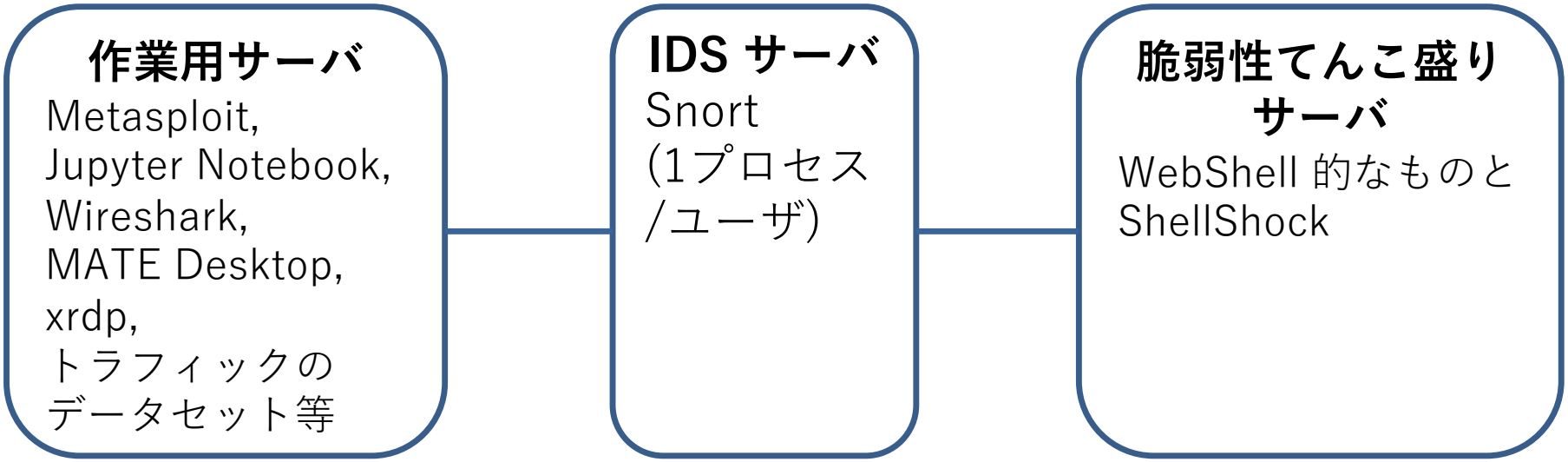
→ トラブルが起こったらその場でなんとかする。
アカウント管理は手を抜く。
環境構築をできる限り自動化し、自動化するために必要なスクリプトのみ保管。毎年使い回す。

不採用の選択肢

- 各々の PC に VM のイメージをダウンロードさせる
  - 十分な性能の PC を持っているとは限らない
    - 本学の演習用端末も性能不足
  - 予めダウンロードするのを忘れた人が WiFi で一斉にダウンロードするのに必要な時間を考えると難しい
    - 演習2日しかないのに
- 本学プライベートクラウドを利用
  - API が提供されず、VM やネットワークの作成を自動化しづらい
  - 年額課金しかなくお金の無駄
  - (閉じたネットワークを利用するのに交渉が必要そうだがそうしてまで利用する必要性はない)
- 物理サーバを買う
  - 仮想化基盤を維持し続ける手間を無視できない
  - 受講生が増えた際にスケールアウトしにくい
  - (補助金が終了する時期がちょうど買い換えの時期)

演習環境

- 3〜4人1グループで以下のVMを提供
  - 受講人数が増えたらこのセットを増やしていく



- 構築時はインターネットに接続、演習時は遮断

今年は約30VM

演習環境全体

SSHログイン

インターネット

Web経由の
GUIアクセス

SSHログインサーバ

VDIサーバ (Guacamole)

管理
サーバ

作業用サーバ

作業用サーバ

SSH
ログイン

全サーバ

IDSサーバ

IDSサーバ

…

NAPT
ルータ

脆弱性てんこ盛りサーバ

脆弱性てんこ盛りサーバ

環境構築の自動化: Infrastructure as Code 向けのツールの採用

- Infrastructure as Code:
  “自動化、バージョン管理、テスト、継続的インテグレーションといった、ソフトウェア開発のプラクティスをシステム管理に応用するための方法論”
  Infrastructure as Code (O’Reilly) より

- 今回は短期間で利用するため、自動化の部分を利用
  - バージョン管理、継続的インテグレーションは重要ではない
  - テストは手つかず

- 理想: コマンド1〜2発で構築完了

環境構築の自動化: VM やネットワークの作成

- VM や閉じたネットワークの作成 → Terraform
  - サーバやディスク、ネットワークの数や接続関係などの宣言的な記述から IaaS の API を利用し自動で作成

```
resource "sakuracloud_disk" "work" {
    count = "${var.set_count}"
    name = "work${count.index}"
    hostname = work${count.index}"
    source_archive_id = "${image_id}"
    plan = "${var.work_disk_plan}"
    size = "${var.work_disk_size}"
    password = "${var.root_password}"
}

resource "sakuracloud_switch" "mgmtsw" {
    name = "mgmtsw"
}
```

```
resource "sakuracloud_server" "work" {
    count = "${var.set_count}"
    name = "${var.work_name}${count.index}"
    disks =
["${sakuracloud_disk.work.*.id[count.index]}
"]
    core = "${var.work_core}"
    memory = "${var.work_memory}"
    nic = "${sakuracloud_switch. mgmtsw.id}"
    ipaddress =
"${var.work_mgmt_ipaddr[count.index]}"
    nw_mask_len = "${var.mgmt_masklen}"
    gateway = "${var.vpc_ipaddr}"
    additional_nics =
["${sakuracloud_switch.loginsw.id}",
"${sakuracloud_switch.workidssw.id}"]
}
```

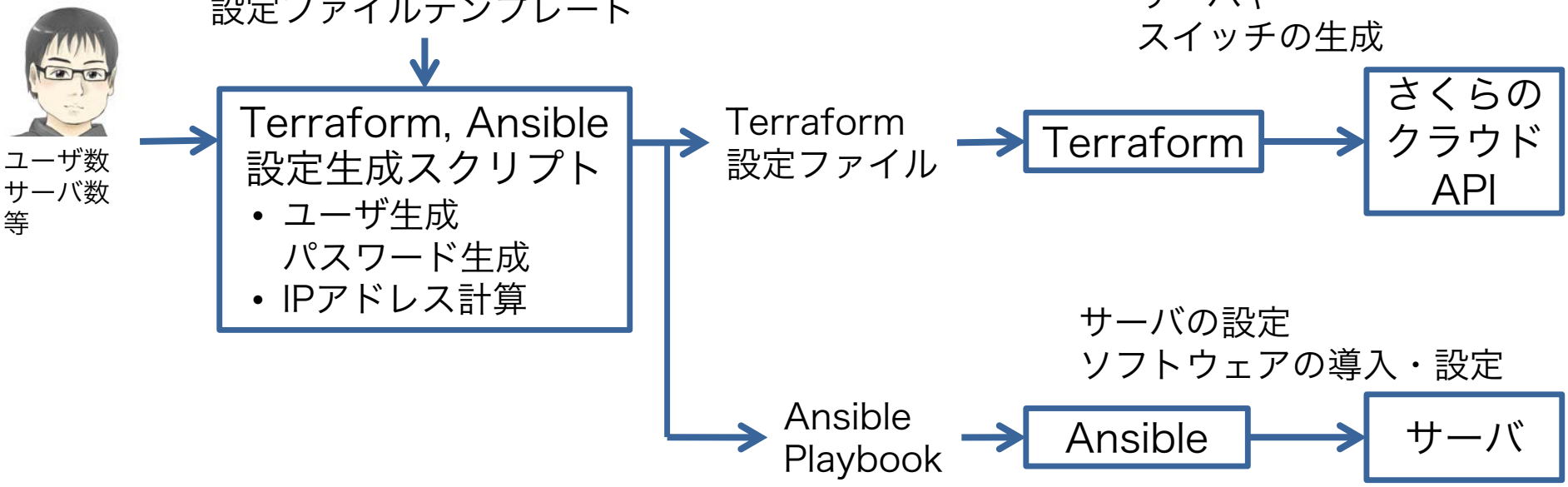
環境構築の自動化: サーバの設定

- ソフトウェアのインストール・設定 → Ansible
  - サーバの構成を管理するツールの1つ
  - サーバにインストールされているべきソフトウェアや設定を記述し実行すると、現状との差分を検出しインストール・設定

```
- name: users exist
  user:
    name: "{{item.name}}"
    password: "{{item.password}}"
    group: "{{item.group}}"
    uid: "{{item.uid}}"
    state: present
    shell: /bin/bash
    groups: seccap
  with_items: "{{users}}"
  become: true
```

```
- name: apt install
  become: true
  apt:
    name: "{{item}}"
    state: present
  with_items: "{{apt_install}}"
```

Terraform + Ansible による演習環境の自動構築



Terraform, Ansible 設定生成スクリプト、Terraform、Ansible はそれぞれ手動実行
（構成を変更したときにその部分だけ再実行しやすいように）

謝辞: 多くの部分を SRC Software さんに外注

狙い通りの効果が得られた点

- 担当者の負担が減った
  - サーバの準備のための単純作業の繰り返しによる精神的な負担
  - うまく動かない時に気軽に再インストールできるようになった
    - 10分ほどあれば終わる
  - 演習内容自体の改善に使える時間が増えたような気がする

- 演習中に不具合が発覚した場合に速やかに修正できる
  - 修正が必要な部分だけ Ansible の Playbook を書いて実行することで一括して変更できる

思ったより軽くならなかった部分

- Ansible の Playbook で書きにくい処理がいくつもある
  - コンパイルしてインストールする必要のあるソフトウェア
  - GUI で変更することを想定しているような設定
    - 日本語表示・入力にするとか
    - 資料がなかなか見つからない
  - データベースの初期化

- Ansible Playbook の維持に手間がかかる
  - ソフトウェアのアップグレード対応
    - Ubuntu 16.04 LTS → Ubuntu 18.04 LTS: OS の基本的な設定方法の変更 (netplan)
  - Bug fix や演習内容の改善への対応

年1回の演習のためにどこまで手間をかけるのが適切か難しい

継続性の課題

- 価格が抑えられている
  - クラウド利用料: 約2500円/人
    - 準備・確認・アップデートのための利用料を除く
- 特定のクラウドサービスへの依存が課題
  - Terraform から設定するためのプラグインが必要
  - Terraform の設定ファイルの記述がクラウドサービス依存
    (さくらのクラウドがなくなったら他のクラウドに向けて書き直し)
- Ansible Playbook の実行時間が長い
  - フルで実行すると6時間程度 (コンパイルなどしている都合)
  - コピー元となるディスクイメージを作るための Playbook と、
    各サーバ固有の設定をする Playbook を分けたほうがよさそう
  - コンテナの活用？
- 担当者への依存

まとめ

- Basic SecCap の京都大学開講演習科目「情報セキュリティ演習」の演習環境を半自動化
  - Ansible と Terraform を活用
    - 同じような構成の VM を自動で大量に作ることができる
    - 構築スクリプトのみ保存しておけばよく、サイズが小さくて済む
- 自動化により得られる利点と手間のバランスが難しい
  - 年1回しか構築しない環境でどこまで手をかけるか
    - 多少汚くても動けばいいのをある程度許容する
  - テストは手付かず