セキュリティ講座
演習資料

ver 1.0

演習環境の下準備手順

演習 4 と演習 5 でパソコンを使った演習を行うため、実機の環境構築手順を示します。

作業1. VirtualBox のインストール

__1. Oracle VirtualBox を、標準設定のままインストール。

__2. [ファイル(F)]-[ホストネットワークマネージャー(H)...]を開き、 [作成(C)]をクリック。

__3. 新しくできた Host-Only Ethernet Adapter #2 で、以下の設定を行う

アダプター(A)
◉アダプターを手動で設定(M)　←選択
IPv4 アドレス:　192.168.33.1
IPv4 ネットマスク(M):　255.255.255.0
DHCP サーバー(D)
□サーバーを有効化(E)　←チェック無し

__4. [ファイル(F)]-[環境設定(P)...]-[ネットワーク]で、右側の「新しい NAT ネットワークを追加します。」ツールボタンをクリックして追加する。

作業2. 仮想マシンのインポート

__1. [ファイル(F)]-[仮想アプライアンスのインポート(I)...]

__2. pen_training.ova を選択し、そのまま[インポート]

作成された 2 つの仮想マシンのうち、Kali-Linux の USB 設定を変更する。

__3. 仮想マシン Kali-Linux を選択し、[設定(S)]をクリック

__4. [USB]-[☑USB コントローラーを有効化(U)]で、[◉USB 1.1 (OHCI) コントローラー]を選択する。

作業3. 動作確認

__1. 仮想マシン Mutillidae をダブルクリックして起動。
起動しない場合、メッセージに従って修正を行ってください。

__2. ホスト PC から以下の ping が届くことを確認。

```
ping 192.168.33.10
```

__3. 以下のアカウントでログインできることを確認。

```
mutillidae login: vagrant
Password: vagrant
```

__4. 仮想マシン Kali-Linux をダブルクリックして起動。
起動しない場合、メッセージに従って修正を行ってください。

__5. 以下のアカウントでログインできることを確認。

```
Username: root
Password: toor
```

__6. ホスト PC から、以下の ping が届くことを確認。

```
ping 192.168.33.11
```

__7. 先にログインした Mutillidae のコンソールから、以下の ping が届くことを確認。

```
ping 192.168.33.11
```

__8. 仮想マシン Mutillidae と Kali-Linux をシャットダウンし、作業終了。

演習1. 情報資産と脅威の検討

どんなものが情報資産となるか改めて認識したうえで、脅威、脆弱性、リスクをグループで検討していきます。最後にリスクについて検討し、全体で共有を行います。

作業1. 個人作業（1 分）

__1. （差支えない範囲で）自組織で守るべき情報資産を 1 つ書き出してください。

作業2. グループ作業（2 分）

__1. 守るべき情報資産をグループメンバに提示してください。

作業3. 個人作業（7 分）

__1. 各情報資産に対する脅威を 3 つ以上列挙してください。

作業4. グループ作業（15 分）

__1. 脅威についてグループ内で全体共有し、脆弱性とリスクを検討してください。

__2. 見落としがち、または対策が難しそうなリスクを 1 つ選び、全体で対策を検討する。

作業5. 全体共有(5 分)

__1. 選んだ脅威と対策検討結果を発表していきます。対策は完全でなくてかまわないので、どのような検討を行って、どのような点が問題なのか伝えるようにしてください。

演習2. インシデント対応事例 - 正当なアカウントによる侵害

以下のケーススタディでどのような行動をとるべきか。節目節目で考えます。
時間は目安です。演習時間が限られているため、時間内にできたところまでで演習結果をまとめ、グループ間で発表を行います。各グループの進捗の違いも今後の参考となります(なぜ早く進んだのか、なぜ時間がかかったのか、など)。

作業1. 検知と連絡受付、トリアージ (15 分)

1. あなたの組織のセキュリティスタッフが、セキュリティ情報およびイベント管理ツール (SIEM: Security Information and Event Management)のテスト運用を開始しました。VPN のログも監視対象に含めたところ、アカウント Akira が数日間にわたり複数のシステム、複数の IP アドレスから同時に VPN ログインしていることが検知されました。

__1. セキュリティスタッフのメンバーがこの時点ですぐに行わなければならないことは何でしょうか。グループ内で話し合って 3 つほど挙げ、優先順位をつけてください。

作業2. インシデント対応 – 初動、調査（45 分）

2. アカウント Akira を無効化したところ、すぐにアカウント Ibuki による同様のアクセスが検知されました。セキュリティスタッフは経営陣と CSIRT に状況を報告することにしました。そしてあなたは CSIRT のメンバーとして**初動対応**をとることになりました。

初動対応の流れを簡単に示します。慣れている方は読み飛ばして構いません。

インシデント対応の主な活動は 3 つあります。実際には、組織全体の利害関係者間で活動内容について文書化されている必要があります。

1. 初動
   - 対応チーム編成、データの検討、インシデント見極め
   - 適切な対応を判断できるだけの情報を集める
2. 調査
   - どういう経緯で何が起こったのか
   - 責任の所在
3. 修復
   - 修復計画は、インシデント対応の初期段階で検討
   - 法律、ビジネス、政治、技術面を考慮
   - 修復のタイミング
     - 脅威の存在を示す痕跡がなくなったタイミングで実施

上記 3 つの主な活動内容の例をざっと以下に示します。

1. 初動～一般的な活動～
   - セキュリティスタッフに対する、状況の聞き取り調査
   - 状況を技術的に正しく判断できる IT スタッフに対する聞き取り調査
   - インシデントに関連すると思われる事業部門の職員への聞き取り調査
   - ネットワークログやセキュリティログによる、インシデント発生の裏付け
   - 集めたすべての情報の文書化

2. 調査～次の 5 段階で調査を実施～
    1. 最初の手がかり
        - インシデントと関連性があるか
            - 進行中のインシデントと関係なければ除外する
        - 具体的かつ詳細か
            - 5W1H を意識する
        - 利用できる手がかりか
            - 関連性があっても原因につながらない手がかりもある
    2. 脅威が存在することを示す痕跡（IOC[1]: Indicator of Compromise）
        - 作業ディレクトリ名、出力ファイル名、ログインイベント、永続化メカニズム（データベース）、IP アドレス、ドメイン名、マルウェアのプロトコル上の特徴など
        - インシデント検出の自動化でも使用可能
        - ホストベースインディケーター
            - シグニチャーベース
            - 方法論ベース
        - ネットワークベースインディケーター
    3. 疑わしいシステムの特定
        - 検証
            - そのシステムは本当に疑わしいシステムか。
        - ラベル付け
            - バックドア導入、データ窃盗、認証情報収集、SQL インジェクション、など
        - 優先順位付け
    4. 証拠の保全
        - システム操作時間は最小限で。システム上の変更は極力避ける。適切に文書化する。
        - ライブレスポンス（稼働中のシステムの情報収集）
            - プロセスリスト、稼働中のネットワーク接続、イベントログ、レジストリなど
        - メモリー収集[1]（可能な場合）
        - フォレンジックディスクイメージ[1]

5. データ解析
    - マルウェア解析
        - 疑わしいプログラムのトリアージ
        - マルウェアの基本機能の洗い出し
        - リソースと予算があれば、疑わしいマルウェアの動作解析
    - ライブレスポンス解析
        - システムへの不正アクセスによる影響の把握
        - 些細な見逃しが致命的となることもある
    - フォレンジック解析
        - 疑惑の裏付け、または否定する情報を発見する
        - 攻撃シナリオを想定し、解析範囲を絞り込む
            - 限られた時間の使い方を強く意識する

3. 修復～重要な調査情報をまとめる～

修復の計画は３つに分けて考えます

1. 態勢整備
    - 修復を確実に成功させるため、段階を踏んだ手順
    - 手順策定、連絡先の交換、責任の明示、可視化、リソース確保、スケジュール調整など
2. 短期的戦術
    - 今起こっているインシデントに対する対処
    - システム再構築、パスワード変更、パケットフィルタリング、広報、業務処理の変更
3. 長期的戦略
    - 情報セキュリティ管理体制の見直し

そして、重要な調査情報をまとめておきます。

- 集めた証拠の一覧
- 影響を受けたシステムの一覧
- 疑わしいファイルの一覧
- アクセスされた、または窃取されたデータの一覧
- 攻撃者による重大な活動の一覧
- ネットワークベースの、脅威の存在を示す痕跡
- ホストベースの、脅威の存在を示す痕跡
- 侵害を受けたアカウントの一覧
- 進行中のタスクと、処理すべきタスクの一覧

ここから作業 2 の演習手順です。

__1. 職員からの聞き取り結果の一部を下方に示します。ここからどのような状況が読み取れるか、ほかにどのような情報が必要かをグループ内で検討してください。そして、**調査の手がかり**となる項目を検討し、列挙していってください。

__2. 列挙した調査項目の優先順位を決めてください（高・中・低くらいの粒度で）。

__3. 優先順位の高い調査項目から、継続中の攻撃を排除または修復する方法を考え、まとめます（不完全で構いません）。

職員からの聞き取り結果の一部：

- 組織内の複数の部署で標的型攻撃メールを約 3 か月前に受信。合計１００通。PDF ファイルが添付されていた。受信者の中に Akira は入っていたが、Ibuki は入っていなかった。Akira を含め何人かが添付ファイルを開いたが、感染が確認されたのは Akira だけである。
- 標的となったメールアドレスは、先日開いた技術カンファレンスの講演者とその関係者のアドレスに限られていた。
- 技術カンファレンスでは、研究を進めている画期的な新技術について紹介を行った。その研究の機密データは別途のサーバーSV01 に保管され、機密プロジェクト関係者以外が読み書きできないよう、適切にアクセス制御設定がされている。
- 通信ログから、 1 か月前にサーバーSV01 から外部 FTP サーバーへ cab 形式ファイルが送信されていることは確認できた。暗号化されていたため cab ファイルの内容はわからない。機密データは削除されていない。
- 各職員のアカウントには、各自で使用する PC に対するローカル管理者権限が与えられていた。
- システム管理者の利便性から、ローカル管理者である administrator のパスワードはすべてのサーバーとクライアントで共通化されていた。
- Akira のパソコンから Gh0st RAT (Remote Access Tool)が発見されたが、RAT の通信は件のメール受信後数日しか観測されていない。
- Akira は VPN 経由で自宅から組織内のネットワーク環境に接続していた。

作業3. インシデント対応 – 調査 – 脅威が存在することを示す痕跡（15 分）

3. 攻撃の脅威は今も続いています。脅威があることを裏付けし、修復後に脅威がなくなったことを保証するために、**脅威が存在することを示す痕跡、インディケーター(IOC)**を明確にしなければなりません。

__1. インディケーターは手がかりを根拠に決定していきます。この事例では、どのような情報がインディケーターとして使えるか、グループ内で検討してください。インディケーターの例をいくつか示します。

- **攻撃元 IP アドレスと接続先ポート番号**
- **サーバーSV0 における cab ファイルの作成イベント**
- **など**

作業4. インシデント対応 – 調査 – 疑わしいシステムの特定、証拠の保全 (15 分)

4. IOC のインディケーターを基に脅威の検知を行ったところ、ほかのコンピュータからも依然としてビーコンが送出されていることがわかりました。外部 FTP サーバーへの接続はサーバーSV0 のみであることも確認できました。攻撃者が使用しているアカウントは、今のところ Akira と Ibuki のアカウントだけです。アカウント Akira による VPN 接続には、Akira の家からの接続も含まれています。組織内で使用されているサーバーは 10 台、クライアント数は 1000 台あります。CSIRT では、インシデントに巻き込まれた**疑わしいシステムを特定**し、**証拠の保全**対象にするコンピュータを選ぶ必要があります。

インシデント対応ではほとんどの場合、すべてのコンピュータの証拠保全を行う必要はなく、また保全を行う時間も解析する時間もありません。

__1. 証拠の保全対象となるコンピュータを次の中から選んでください。保全対象となる理由、ならない理由をグループ内で検討し、納得できるようにしてください。

1. **職場にある Akira のコンピュータ**
2. **職場にある Ibuki のコンピュータ**
3. **家にある Akira のコンピュータ**
4. **サーバーSV0**
5. **10 台のサーバすべて**
6. **ビーコンが送出されているコンピュータ**
7. **すべてのクライアントコンピュータ**

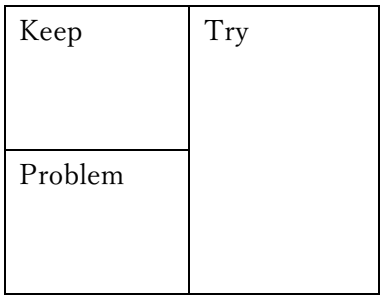
作業5. インシデント対応 – 調査 –証拠の分析（20 分）

5. 保全した**証拠を分析**した結果、以下の事実が分かりました。

- Akira のコンピュータから以下の情報が攻撃者に知られた可能性がある
- ユーザー名、パスワード、クライアント証明書、ローカル管理者パスワード
- Akira のコンピュータの PDF リーダーのバージョンが古いままであった
- ローカル管理者パスワードを使い、SV0 に侵入。
- Ibuki のアカウントは SV0 から入手したと思われる。
- 攻撃者による内部偵察は 3 週間に及んでいたこと。
- SV0 の機密データは暗号化 RAR ファイルとして圧縮し、拡張子を cab に変換し、FTP 送信していた。
- 送信後に RAR ファイルを削除し、Windows のデフラグツールを使用して削除データの復元を困難にしていた。
- VPN 接続に使用された IP アドレスの 1 つが、Gh0st RAT のビーコン送付先と一致していた。その IP アドレスは、プロバイダーとしても、地域としても、Akira とは全く無関係であった。
- 暗号化 RAR ファイルと VPN 接続の影響で、具体的にどのような機密が漏洩したかは突き止められなかった。

__1. CSIRT では修復作業を行うことにしました。修復計画の短期的戦術として、今起きているインシデントに速やかに対処するにはどのような対策をとればよいでしょうか。グループで検討し、対策の例を列挙してください。

作業6. ふりかえり – KPT (Keep-Problem-Try)法 (30 分)



__1. 模造紙で Keep/Problem/Try の枠をつくります。

__2. 本演習で気づいたことのうち、今後も続けていきたいことを各個人で付箋紙に書き、Keep 欄に貼り付けます。

__3. 本演習を通じて問題となったこと、改善が必要と感じたことを各個人で付箋紙に書き、Problem 欄に貼り付けます。

__4. グループで Keep 欄と Problem 欄を整理してください。そして、Keep を改善する案や Problem を解決する案を（すべてでなくてよいので）付箋紙にまとめ、Try 欄に貼り付けます。

__5. 出来上がった模造紙を用い、Try 欄から特に重要と思われるものを 3 つ選び、全体で発表を行います。

演習3. セキュアシステム、ネットワークの設計～脅威モデリング～

脅威モデリングが効果的であることはセキュアシステム開発で実証されつつあります。しかし脅威モデリングでは、どのような脅威があるのかをすべて洗い出すのが難しく、脅威がどのような場合に顕在化するか明確にすることに時間がかかります。そこで、脅威モデリングと対になるセキュリティ要件をシステム上で考えることで、モデリングの負担を減らすことも考えられています。脅威の対策はセキュリティ要件とほぼ同義ですが、幅広いセキュリティの知識と経験から脅威を洗い出す脅威モデリングよりも、具体的なシステムに限定して必要な要件を洗い出すほうが手続きとしては取り組みやすくなります。

ここでは渋滞ナビシステム構築を考えてみます。このシステムではスマートフォンやカーナビのナビアプリで取得した交通状況をサーバー上に蓄積し、走行地区ごとの渋滞情報をナビアプリに配信するサービスを提供するものとします。

まずは脅威モデリングに倣い、セキュリティ要件を整理します。
コンピューター上の事象は次の 3 段階で考えられます。

1. 『誰が』
2. 『どの情報/資産に対して』
3. 『読み/書き/実行するのか』

これらを以下の用語で表現することにします。

1. 『アクター』
2. 『資産』
3. 『操作』

作業1. アクターの洗い出し（10 分）

本ナビシステムのアクターとして考えられる登場人物を挙げ、分類してください。これはグループ内で検討し、共有します。

作業2. 資産の洗い出しと操作（20 分）

個人作業で行います。

本ナビシステムの資産として考えられるものを挙げていきます。その際、「意図しない人に勝手に使われては困る情報や機能」に限定していくつか列挙してください。すべてでなくてかまいません。そして、洗い出した資産に対し、主要なアクター（1～2 アクターでよい）に許可される操作を列挙してください。

セキュリティ要件を洗い出していくと、いくつかのパターンが見えてきます。このパターンを類型化することで対策を単純化できます。簡単なケースとして、『「攻撃者」は「○○資産」を読めない』というルールはかなり頻繁に登場するのでまとめることができるでしょう。

例：

資産	セキュリティ要件
2 地点間の通過所要時間 ├開始地点と終了地点 └通過所要時間（秒）	「利用者」は、「所要時間」を読める、書けない 「ナビアプリ」は、「所要時間」を読める、書ける 「第三者」は、「所要時間」を読めない、書けない
2 地点間通過速度の揺らぎ ├開始地点と終了地点 └速度の偏差	「利用者」は、「揺らぎ」を読めない、書けない 「ナビアプリ」は、「揺らぎ」を読める、書ける 「第三者」は、「揺らぎ」を読めない、書けない
特定地点の道路状況取得	「利用者」は、「道路状況取得」を実行できる 「ナビアプリ」は、「道路状況取得」を実行できる 「第三者」は、「道路状況取得」を実行できる
…	…

※ 一つの資産が複数の資産の集合体であることもあります

※ 本来は物理的資産も考えますが、ここでは情報資産に限定して構いません。

※ 「攻撃者」が○○機能の「妨害」を実行できない、というセキュリティ要件もあります。資産に対するセキュリティ要件は、ここでは自由に考えてください。

資産	セキュリティ要件

※ 本作業で洗い出したセキュリティ要件はそのまま脅威モデリングの脅威と対応します。

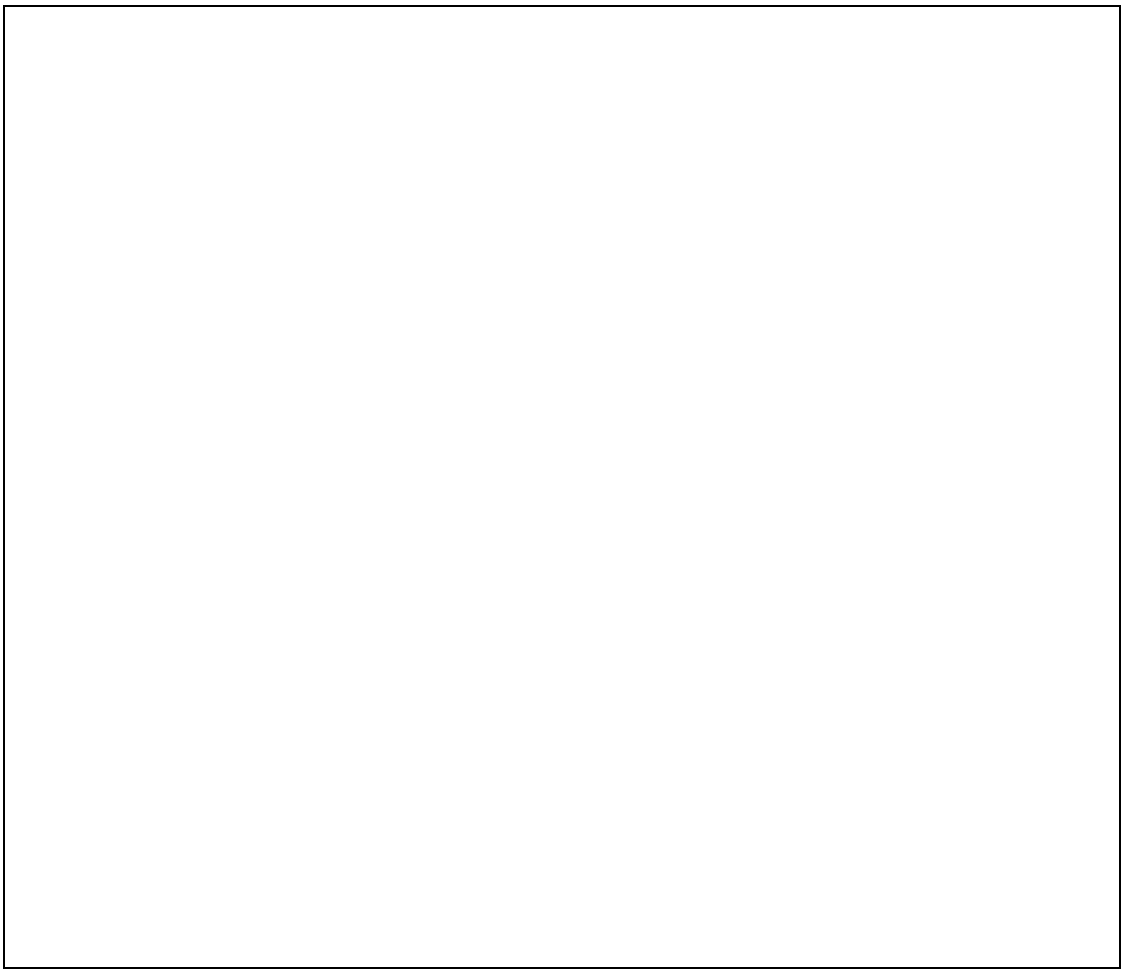
作業3. 資産の所在（20 分）

個人作業で行います。

多くの情報資産は、ネットワークを介してクライアントとサーバー間を行き来します。そのすべての経路において脅威が存在します。本件の渋滞ナビシステムではどのようなソフトウェア、装置、経路に脅威が存在するでしょうか。

想定されるネットワークの概略図を描き、先の（情報）資産に対する脅威の存在する箇所、すなわちセキュリティ要件を確認すべき個所を考え、脅威の例をいくつか書きこんでください。

※ 本来なら、どの情報資産に対する脅威かを明らかにすべきですが、ここでは脅威だけをあげていただければ構いません。



○をつけたすべての個所でセキュリティ要件が満たされたならば、それはセキュアなシステムであることになります。

作業4. グループディスカッション（20 分）

作業２と３の結果をグループ内で共有し、どのようなセキュリティ要件が存在し、どの箇所で検証すべきかを共有及び検討してください。気づかなかったアイディアや構成、セキュリティ要件があった場合は記録しておきます。

演習4. 手動による Web アプリケーションの脆弱性チェック

ここでは OWASP で提供する OWASP Mutillidae（ミューティリデ） 2 という、意図的に脆弱にした Web アプリケーションを用い、Web アプリケーションの脆弱性の体験とテストを行います。
最初は手動で脆弱性を探し、引き続き脆弱性チェックツールを使ってみます。
演習は個人ごとに行えますが、攻撃手法やチェックツールの検査結果の読み取り方でぜひディスカッションを行ってください。
なお、オプション演習は後回しにし、時間に余裕がある場合に実施してください。

作業1. 脆弱な Web アプリ Mutillidae の起動と表示（5 分）

__1. VirtualBox マネージャーを起動します。
__2. 仮想マシン Mutillidae をダブルクリックして起動します。ログインはしません。
__3. VirtualBox マネージャーが起動するホスト PC でブラウザーを開き、以下の URL にアクセスします。

```
http://192.168.33.10/mutillidae/
```

作業2. SQL インジェクション（10 分）

最初に SQL インジェクションのテストを行います。

__1. 以下のメニューを選びます。
[OWASP 2017]-[A1 - Injection (SQL)]-[SQLi - Extract Data]-[User Info (SQL)]
__2. 以下のデータ入力後に[View Account Details]をクリックし、通常動作を確認します。

```
Name        admin
Password    adminpass
```

問題 1.

このページには SQL インジェクションの脆弱性が含まれています。
[Hints and Videos]も参考にしつつ、以下の確認をしてください。
必要であれば、インターネットの翻訳サイトで英語を翻訳してください。

1-1) SQL インジェクションの脆弱性があることの確認
1-2) 様々な入力を試して表示されるエラーメッセージから、どのような内部処理をしているか推察してください。
1-3) 脆弱性を用い、全ユーザー情報を閲覧できることを示す
1-4) （オプション）試行錯誤して、データベース名と SQL サーバーへのログインユーザー名、バージョンなどの取得を試みてください。

作業3. コマンドインジェクション（10 分）

コマンドインジェクションを試します。

__1. 以下のメニューを選びます。

[OWASP 2017]-[A1 - Injection (Other)]-[Command Injection]-[DNS Lookup]

__2. （オプション)ホスト PC がインターネットにつながっている場合、Hostname/IP 欄で幾つか動作を確認してください。

問題 2.

このページにはコマンドインジェクションの脆弱性が含まれています。

[Hints and Videos]も参考にしつつ、以下の確認をしてください。

2-1) /etc/passwd を表示させる

2-2）内部の IP アドレスを調査する

2-3）その他、ディストリビューション確認やカーネルバージョン確認など。

作業4. （オプション）認証の不備（10 分）

SQL インジェクションの脆弱性を用い、認証の不備を試します。

__1. 以下のメニューを選びます。

[OWASP 2017]-[A2 - Broken Authentication and Session Management] -[Authentication Bypass] -[Via SQL Injection]-[Login]

__2. 以下のデータ入力後に[Login]をクリックし、ログイン失敗を確認します。

```
Name        admin
Password    password
```

__3. 以下のデータ入力後に[Login]をクリックし、通常動作を確認します。

```
Name        admin
Password    adminpass
```

__4. [Logout]でログアウトします。

問題 3.

[Hints and Videos]も参考にしつつ、ユーザー名もパスワードも分からない前提で、SQL インジェクションを用いてログインしてください。

作業5. （オプション)機微な情報の露出 （10 分）

機密情報ではないですが、機微な情報として PHP 情報を取り出してみます。

__1. 以下のメニューを選びます。

[OWASP 2017]-[A3 - Sensitive Data Exposure]-[Information Disclosure] -[PHP Info Page]

問題 4.

以下の情報を確認してください。

PHP Version _____

OpenSSL Library Version _____

作業6. XML 外部エンティティ参照 (XXE) (10 分)

XML 外部エンティティ参照 (XXE)では、XML 検証ページに不正な XML を入力し、スクリプトを実行させてみます。

__1. 以下のメニューを選びます。
[OWASP 2017]-[A4 - XML External Entities]-[XML External Entity]-[XML Validator]

__2. [Hints and Videos]-[XML External Entity (XXE) Injection]を開き、[Videos]のすぐ上に例示されている以下の XML(改行は任意)を入力し、[Validate XML]をクリックします。

```
<?xml version="1.0"?>
<change-log>
  <text>
    &lt;script&gt;alert(&quot;Hello World&quot;)&lt;/script&gt;
  </text>
</change-log>
```

__3. <text>要素に記述されているスクリプトは実行されましたか(はい ・ いいえ)。

残りのペネトレーションテストは、時間に余裕があるときに試してみてください。

手動で行うペネトレーションテストは時間がかかることは実感できたはずです。また、脆弱性を確認するだけであれば、具体的な攻撃方法を考える必要もありません。例えば SQL インジェクションの脆弱性チェックでは、シングルクォーテションの入力だけでも確認できます。

次の演習では、Kali Linux に同梱されているペネトレーションテストツールを用いた脆弱性検査を幾つか体験します。

演習5. ツールを使った Web アプリケーションの脆弱性チェック

作業1. 下準備 (5 分)

__1. VirtualBox で、Kali-Linux をダブルクリックして起動します。

__2. 以下の情報でサインインします。

```
Username: root
Password: toor
```

引き続き環境を日本語化します。

__3. デスクトップ上部のメニュー右端の▽を開き、メニュー左下の設定ツールをクリックして開きます。

__4. Region & Language で、日本語表示設定をします。

__5. Restart をクリックし、サインインしなおしてください。

作業2. OWASP ZAP (10 分)

OWASP が提供する OWASP ZAP は、操作が簡単なペネトレーションテストツールです。まずはこのツールを試してみます。

__1. [アプリケーション]-[03 - Web Application Analysis]-[owasp-zap]を実行します。Apache License は[Accept]してください。

__2. Do you want to persist the ZAP Session? では、一番上のラジオボタンにチェックを入れて、[Start]します。

__3. [Tools]-[Options…]-[Language]で日本語にし、owasp-zap を開き直します。

__4. ZAP セッションの保持方法をどうしますか？ では、一番上のラジオボタンにチェックを入れて、[開始]します。

__5. クイックスタート タブの攻撃対象 URL に以下を入力し、[攻撃]をクリックしてしばらく待ちます。スキャンが終了したか否かは、[停止]ボタンの状態で判断できます。

```
http://192.168.33.10/mutillidae/
```

問題 1. スキャン終了後の以下の項目を確認してください。

1-1) OS コマンドインジェクションが見つかったページの PHP ファイル名

/index.php?page=_____

1-2) SQL インジェクションが見つかったページの PHP ファイル名（一部で構いません）

/index.php?page=_____

1-3) パストラバーサルで、気になる URL を試してください。
1-4) その他、見つかった脆弱性で気になるものがありましたら確認してください。

作業3. Nikto2 (10 分)

引き続き、Web サーバーのスキャナである Nikto を試します。Nikto はオープンソース(GPL)で、できるだけ短い時間にテストを行うように設計されています。

__1. [アプリケーション]-[02 - Vulnerability Analysis]-[nikto]を実行します。

__2. 以下のコマンドで Nikto によるスキャンを実行します。

```
nikto -host http://192.168.33.10/mutillidae/ -o /tmp/output.txt
```

問題 2. 結果ファイル/tmp/output.txt を見て(less /tmp/output.txt)、以下の確認をしてください。

2-1) クリックジャッキング対策はされていますか。　　(はい ・ いいえ)
2-2) ディレクトリ一覧が有効な URL はどこですか。いくつか列挙し、その URL の動作を確認してください。

/multidae/_____
/multidae/_____
/multidae/_____
/multidae/_____

2-3) HTTP の TRACE メソッドは有効ですか。　　(はい ・ いいえ)

※ クリックジャッキング：リンクやボタンを偽装して、利用者の意図しない動作をさせようとする手法。

作業4. Skipfish (オプション)

最後に、Google が提供するセキュリティチェックツールである Skipfish の使い方の一例を紹介します。本演習環境で実施すると、チェックにほぼ半日かかります。もし実行する場合、途中で停止（Ctrl + C）して結果を確認するようにします。

__1. 仮想マシン Mutillidae に、以下のアカウントでサインインします。

```
mutillidae login: vagrant
password: vagrant
```

__2. DB サーバーmysql と Web サービス apache2 を再起動します。

```
sudo service mysql restart
sudo service apache2 restart
```

※ ここまでの作業でサービスが落ちたり止まったりしている場合があるため。

__3. 使用状況確認のため、top コマンドを実行しておきます。

```
top
```

__4. 仮想マシン Kali-Linux で、以下を実行します。
[アプリケーション]-[03 - Web Application Analysis]-[skipfish]

__5. 表示された端末で、以下のコマンドを実行してスキャンします。

```
touch /tmp/test.wl

skipfish -W /tmp/test.wl ¥
-S /usr/share/skipfish/dictionaries/minimal.wl ¥
-o /tmp/result ¥
-auth-user admin ¥
-auth-pass adminpass ¥
http://192.168.33.10/mutillidae/
```

__6. なにがしかのキーを押してスキャンを開始します。

__7. 途中で止める場合、Ctrl+C で止めてください。それまでのスキャン結果は保存されます。

__8. 以下のコマンドで実行結果を表示します。

```
firefox /tmp/result/index.html
```

実行結果の見方に関するドキュメントはありません。赤い丸や赤い旗の項目を展開し、trace を見ることで、どのような脆弱性が発見されたか確認できます。

演習6.　コンプライアンス事例の検証

ここではあるシステム障害事例を基に、コンプライアンスの観点でどのような問題があったか考えていきます。

作業1.　銀行の統合に伴うシステム障害事例（10 分）

まずは以下のシステム障害の経緯と結果を確認してください。

銀行の統合にあたり、３銀行の勘定系システムを統合することになった。

11 年 12 月（残り 2 年 4 か月） 3 銀行のシステムをＩ社のシステムで一本化し、14 年 4 月 1 日に稼働させることを計画。
12 年 12 月（残り 1 年 4 か月） 3 銀行の頭取による政治力学的な事情により、Ｉ社、Ｆ社、Ｈ社の既存システムをリレーコンピューターでつなぐ方式に計画を変更。
13 年 3 月～6 月　（残り約 1 年） 金融庁よりスケジュールの遅れを指摘される。指摘されたスケジュールの遅れは担当部署からたびたび報告され、経営陣も把握しているものの、適切な指示も詳細な調査も行われなかった。
13 年 5 月（残り 11 か月） 追加システムの特殊なバグ（後日判明）がテストで発見できないまま、３銀行のシステム移行計画が持ち株会社の取締役会で決定される。
13 年 12 月　（残り 4 か月） Ｈ 社のシステムが大手取引先の大量データ処理に向いていないことが分かり、Ｆ 社のシステムを引き継ぐことに計画を変更。同時に金融機関コード、店番号コードの取り扱いを変更。
14 年 3 月上旬　（残り 1 か月） 口座振替の強化テストを実施。ただし時間的余裕がないことから、口座振替データ誤入力時のシステム全体の負荷を調べる異常テストは行われなかった。
14 年 3 月 22 日（残り 10 日） 直前の強化テスト実施後、同月 22 日の経営会議で、システム担当部署からの「おおむね問題なく進んでいる」との回答により、それ以上の確認はせずシステム移行を決定した。異常テストが行われなかったことは、ここでは報告されなかった。
14 年 3 月 25 日の週 のちに語られたＩ社担当者の言によると、3 月 25 日の週の時点でシステム完全統合のめどは立っておらず、4 月 1 日の稼働は絶望的であることが分かったと明かす。
14 年 3 月 31 日（前日）

口座振替データの入力を開始したが、新旧の「金融機関コード」「店番号」が混りミスマッチ発生。
14 年 4 月 1 日（当日） システム稼働を強行。大規模障害発生 のちの語られた F 社担当者の言によると、システム統合作業は 3 月 31 日夜に終わったため、ATM の試験がほとんどできなかったと明かす。

主な結果は以下の通り。

- ATM 障害発生、口座振替は、手作業で修正を続けたが、約 10 万件が未処理
- 顧客の口座データの消失等の二次被害も発生、収束までに約 1 ヶ月を要した
- 金融庁は日銀と共同で持ち株会社に対し 1 カ月立ち入り検査を実施し、業務改善命令を出す
- システム障害に伴う損害は 18 億円程度と発表
- 現場担当者の処分は無し。管理職以上は社内規定に従い処分。情報統括役員は降格の上、辞任。全役員 117 名を減給処分。
- 最終的に 18 年 12 月にシステム統合完了を宣言。しかし 23 年 3 月 14 日、想定外の大量振り込み処理を、統合した F 社の旧システム(01 年より稼働)が対応できず障害発生。24 年、勘定系システムの全面刷新・統合プロジェクトを立ち上げる。延期が続くも、31 年末のシステム切り替え完了を目指す。

作業2. 事例の検討 （40 分）

この事例をコンプライアンス遵守の観点から見ると、法令順守の問題ではなく、社内規定の問題です。このようなシステム障害をなくすためにはどのような対策をとればよいか、グループ内で検討し、３個～１０個程度の箇条書きの形でまとめてください。

作業3. 検討結果の共有 （10 分）

問題点と対策の検討結果のいくつかをグループ単位で発表し、全体で共有を行います。基本的には質疑応答なしで構いません。

作業4. 演習　Kali Linux を使った Windows Server 2003 へのペネトレーション

※ Kali Linux に収録されている armitage と Metasploit を使って Windows Server 2003 のぜい弱性を発見し、不正侵入テスト（ペネトレーションテスト）を実施します。

※ 注意：**Kali Linux はぜい弱性検査や攻撃のシミュレーションなどに使用できる強力なツールですが悪用すると不正な操作も可能です。自分の管理していないマシンに対して使用すると、「不正アクセス行為の禁止等に関する法律」（「不正アクセス禁止法」）に抵触するので、外部との接続が遮断されている環境、自分が完全に管理している環境以外での使用は、行わないようにしてください。**

1. KaliLinux 仮想マシンを起動し、root でログインします。（パスワードは toor）
2. Win2003std 仮想マシンを起動し、Administrator でログインしておきます。（パスワードは password）
3. 画面の上（中央よりやや右に）4 つの GUI 画面の選択アイコンがあるので左から 2 つめ
4. Terminal を起動します。（画面下中央のドックから黒い「$」アイコンをクリック）
5. postgresql.conf ファイルを変更します。

```
# vim /etc/postgresql/10/main/postgresql.conf <Enter>  <---(<Enter>キーを押す)
:set  nu  <Enter>  <---(<Enter>キーを押す。行番号を表示させる)
<i>キーを押し-挿入-モードに変更し、63 行目を変更
 63 port  =  5433
      ↓
 63 port  =  5432
<ESC>キーを押して-挿入-モードを抜ける
:wq  <Enter>  <---(<Enter>キーを押す。:wq でファイルに変更を書き込み vim エディタを終了)
```

6. 以下のコマンドで postgresql を起動します。

```
# service  postgresql start <Enter>  <---(<Enter>キーを押す)
[ ok ] Starting postgresql (via systemctl): postgresql.service.
```

7. metasploit 用のユーザ msf とデータベース msf を作成します。

```
# su postgres  <Enter>
postgres@kali:/root$ createuser msf -P -S -R -D
新しいロールのためのパスワード:   msf  <Enter><---(<Enter>キーを押す)
もう一度入力してください：  msf    <Enter><---(<Enter>キーを押す)
postgres@kali:/root$ createdb  -O  msf  msf   <Enter><---(<Enter>キーを押す。-O はオー)
postgres@kali:/root$ exit    <Enter><---(<Enter>キーを押す)
exit
~#
```

8. metasploit の設定ファイルを編集します。（変更前のファイルを保存してから編集）

```
# cd /usr/share/metasploit-framework/config  <Enter><---(<Enter>キーを押す)
# cp  -p  database.yml  database.yml.org  <Enter><---(<Enter>キーを押す)
# vim   database.yml    <Enter><---(<Enter>キーを押す)
<i>キーを押し-挿入-モードに変更
development:
  adapter: postgresql
  database: msf
  username: msf
```

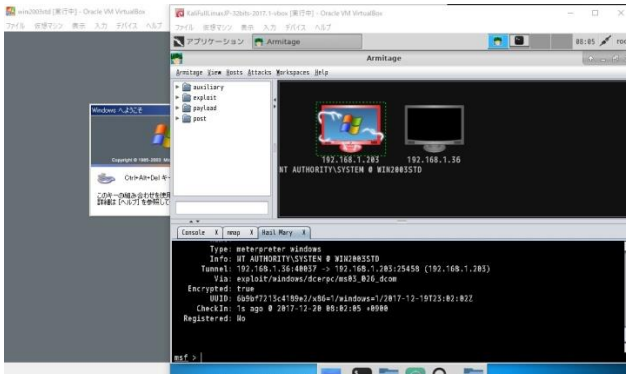
```
  password: hQm2rEM6U3esDNnSCEUPYqKLdfTDgwx+aB/kOfNSVmA=
  以下省略

production:
  adapter: postgresql
  database: msf
  username: msf
  password: msf     <----この行を msf に変更する（これは変更後の状態）
  host: localhost
  port: 5432
  pool: 5
  timeout: 5

test:
  adapter: postgresql
  database: msf_test
  username: msf
  password: hQm2rEM6U3esDNnSCEUPYqKLdfTDgwx+aB/kOfNSVmA=
  以下省略
<ESC>キーを押して-挿入-モードを抜ける
:wq  <Enter>  <---(<Enter>キーを押す。:wq でファイルに変更を書き込み vim エディタを終了)
```

9. GUI を 1 つめの画面に切り替え（4 つの GUI 画面の選択アイコンの左端をクリック）し、。armitage を起動します。(画面左上隅の［アプリケーション］をクリックし「08.Exploitation Tools」＞「armitage」をクリックします。)
10. 「Connect...」ウィンドウが開くので、Pass 欄に「msf」と入力し、[Connect]ボタンをクリックします。
11. 「Start Metasploit?」ウィンドウが開き、Metasploit のサーバを起動するか？と尋ねられるので、［はい］ボタンをクリックします。
12. 「入力」ウィンドウで、「攻撃コンピュータの IP を特定できませんでした。どれですか？」と尋ねられるので 192.168.1.36 と入力し[OK]ボタンをクリックします。
13. メニューバー内の「Hosts」をクリックし、「Nmap Scan」＞「Quick Scan(OS detect)」をクリックします。
14. 「入力」ウィンドウで、スキャンレンジ（範囲）を尋ねられるので「192.168.1.0/24」入力し[OK]ボタンをクリックします。(画面下に nmap タグをもつペインが開きスキャンが開始されます)
15. スキャンが終わると画面上左の Workspace ペインに見つかったマシンが登録されています。(本来表示されないはずの 192.168.1.36 マシンも表示されている)
16. Workspace ペイン内に登録された 192.168.1.203 の Windows マシンをクリックし、メニューバーの「Attacks」をクリックしプルダウンメニュー内の「HailMerry」をクリックします。
17. 「Really？」ウィンドウが開き英語で「一度開始すると HailMerry はワークスペース内のホストにフラッドエクスプロイト攻撃を仕掛けます」と警告されますが、[はい]ボタンをクリックします。
18. スキャンが終わるとワークスペース内に制圧（不正侵入された状態）のホストのアイコ

ンが表示されます。



19. 制圧されたホストのアイコンを右クリックし「Metapreter 番号」>「Explore」>「Browse File」をクリックすると画面下に Windws2003 仮想マシンの C:¥Windows¥System32 内のファイル一覧が表示されます。左上のフォルダをクリックすると現在のディレクトリの 1 つ上のディレクトリに移動します。いろいろ操作して実感してください。
20. 「Metapreter 番号」>「Explore」>「Log keystrokes」を実行し[launch]ボタンをクリックして実行すると、Windows のキーボード入力をキャプチャできます（少し反応が遅い)。Windows でメモ帳を開いて何か入力（アルファベットと数字が分かりやすい）してみてください。
21. 「Metapreter 番号」>「Explore」の他のメニューも実行してみてください。

演習終了