セキュリティハンズオン　#1

Webサービスのセキュリティを攻撃の視点から理解する　~ 入門編 ~

2018.09.28　@kusuwada

注意事項

不正アクセス禁止法違反に注意

今回のハンズオンで学んだことは、
インターネット上のサーバ
自分の管理外のサーバ
に対して絶対に実行しないで下さい！

対象者

- 想定するハンズオン対象者
  - サーバサイド開発・運用メンバー
  - クライアント開発メンバー
  - チームリーダー・プロジェクトマネージャー
- 前提知識・スキル
  - Webサービス・ネットワークの基礎
  - Dockerを触ったことがある
  - Webセキュリティ 初心者

アジェンダ

- はじめに
  - 本日の目的
  - Webアプリケーション・Webサーバの脆弱性
- Webアプリの脆弱性・攻撃手法の紹介と実演
  - XSS（クロスサイトスクリプティング）
  - ディレクトリトラバーサル
  - SQLインジェクション
- おわりに
  - 書籍などの紹介

はじめに

- 本日の目的
  - 開発・運用メンバーに、セキュリティをもっと身近に感じてもらう
  - 「セキュリティ面倒臭い」から「セキュリティ楽しい！」に
  - Webサービスの設計・実装・テスト・インフラ構築・運用時に、セキュリティの視点をプラス

Webアプリケーション・Webサーバの脆弱性

	OWASP Top 10 - 2017 ※1
A1	インジェクション
A2	認証の不備
A3	機微な情報の露出
A4	XML外部エンティティ参照（XXE）
A5	アクセス制御の不備
A6	セキュリティ設定のミス
A7	クロスサイトスクリプティング(XSS)
A8	安全でないデシリアライゼーション
A9	既知の脆弱性のあるコンポーネントの使用
A10	不十分なロギングとモニタリング

※1 Webアプリケーションの脆弱性トップ10 「OWASP Top 10」の2017年版

- 脆弱性・攻撃手法の傾向は年々変化
- OWASP（the Open Web Application Security Project）
  - Web アプリケーションのセキュリティ向上を目的に2001年に設立
  - ソフトウェアのセキュリティに関する情報や技術、プロセスの共有・普及・啓発を行っているオープンなコミュニティ
  - 特定のIT企業等の支配下にはなく、その活動には誰もが自由に参加可能
  - Top Ten 以外にも様々な活動が存在

Webアプリケーション・Webサーバの攻撃手法

攻撃手法	概要
XSS（クロスサイトスクリプティング）	ユーザーのWebブラウザ上で不正なスクリプトを動かす
強制的ブラウジング	リンクされていない秘密情報や管理者機能に直接アクセスする
SQLインジェクション	不正入力により任意のSQL文を実行させる
パラメータの改ざん	Web App の期待する値とは別の値を送信し、誤操作させる
HTTPレスポンス分割	偽ページを多数のに人々に見せる
OSコマンドインジェクション	不正入力により任意のOSコマンドを実行させる
セッション管理に関する脆弱性	セッション情報の推測や盗用を行う
パス／ディレクトリトラバーサル	公開されていないファイルに直接アクセスする
バッファオーバーフロー	範囲を超えた書き込みにより、外部から指定した命令を実行する
バックドアとデバッグオプション	デバッグ用の機能などを悪用して不正な操作を行う

出典：@IT Webアプリにおける11の脆弱性の常識と対策

XSS（クロスサイトスクリプティング）1/4

- 利用者からの入力内容やHTTPヘッダの情報を処理してWebページとして出力するサービスに於いて、Webページの出力処理の脆弱性を利用してWebページにスクリプトなどを埋め込み、偽のサイトを表示させたりCookieを取得したりする攻撃手法
- 攻撃例
  - 本物のサイトに偽のページが表示される
  - ブラウザが保存しているCookieを取得される
  - 任意のCookieをブラウザに保存させられる

XSS（クロスサイトスクリプティング）2/4

- xss_1/readme.md を参考に、環境を構築してみましょう
  - 構築できたら、ブラウザで http://0.0.0.0:5000/ にアクセス
- 自分の名前を入れて、打ち消し線で表示してみよう
- フォントの色を変えてみよう
- 自分のcookieを表示してみよう

XSS（クロスサイトスクリプティング）3/4

- 今回のサイトの対策
  - FlaskFWでは、defaultで入力パラメータのエスケープ処理がされる
    - template.html に `{{ parameter }}` の形式で記述すると、自動的にエスケープ処理がされた状態になります
  - 今回はこれを無効にする処理を挿入しているので、これを除去するだけでOK
    - {% autoescape False %}, {% endautoescape %}
  - 他のWebFWではこういった機能がない場合もある
  - それぞれのWebFW、言語に合ったエスケープ処理をするFW,libraryを導入

XSS（クロスサイトスクリプティング）4/4

- 根本的対策
  - HTTPレスポンスヘッダの Content-Type フィールドに文字コードを指定する
  - Webページに出力するすべての要素に対して、エスケープ処理を施す
  - etc... (Webページの性質によって対策は変わってきます)
- 保険的対策
  - Cookie情報の漏洩対策として、発行するCookieにHttpOnly属性を加える
    - これが設定されたCookieは、HTMLテキスト内のスクリプトからのアクセスが禁止されます
  - XSSの潜在的な脆弱性対策として有効なブラウザの機能を有効にするレスポンスヘッダを返す
    - X-XSS-Protection, Content Security Policy (reflected-xss)

出典: IPA 「安全なウェブサイトの作り方 改訂第7版」

ディレクトリトラバーサル 1/4

- パラメータにファイル名を指定するアプリケーションの脆弱性や、アクセス権限の不備を利用して、ファイルの窃取・改ざん・破壊を行う攻撃手法
- 攻撃例
  - 正常リクエスト
    - GET http://dir.example.com/photos?file=***.jpg
  - 攻撃リクエスト
    - GET http://dir.example.com/photos?file=../../secret.txt

特定のディレクトリ配下にあるイメージコンテンツを取得するAPI

コンテンツディレクトリ以外へのアクセスにより、秘匿情報を取得

ディレクトリトラバーサル 2/4

- directory_1/readme.md を参考に、環境を構築してみましょう
  - 構築できたら、ブラウザで http://0.0.0.0:5000/ にアクセス
- このサイトの何処かに secret.txt があります。探してみましょう。
- このサイトのソースコード hello.py も落ちてます

ディレクトリトラバーサル 3/4

- 今回のサイトの対策
  - ユーザー入力のファイルパス指定が存在する場合はチェック機構を設け、
    入力されたパスが想定されるディレクトリ配下になっているかを確認

```
def image():
    image = request.args.get('image')
    if not image:
        abort(404)
    if not os.path.realpath(image).startswith(os.getcwd()): # 追加！
        abort(404) # 追加！
    return send_file(os.path.join(os.getcwd(), 'static', image))
```

  - ユーザーがpathを直接指定できるIFを用意しない

ディレクトリトラバーサル 4/4

- 根本的対策
  - 外部からの入力値にWebサーバ内のファイル名を直接指定させない
  - ファイルを開く場合は固定のディレクトリ名を指定し、かつ
    ファイル名にディレクトリが含まれないようにする
- 保険的対策
  - Webサーバ内のファイルのアクセス権限の設定を正しく管理する
  - ファイル名のチェックを実施する
    - ※ ../など、特定の文字列を防ぐだけではパターンを網羅しきれないので根本対策
      にはならないことに注意

出典：IPA 「安全なウェブサイトの作り方 改訂第7版」

SQLインジェクション 1/4

- データベースと連携するWebアプリケーションで利用者からの入力情報をもとにSQLを組み立てる場合など、SQLの組み立て方に潜む脆弱性を利用してデータベースの情報の窃取・改ざん・破壊を行う攻撃手法

- 攻撃例
  - 正常ログインリクエスト
    - ID: hogehoge Pass: piyopiyo
  - 攻撃ログインリクエスト
    - ID: admin'-- Pass: (any)
    - もしログイン処理のSQLが　SELECT * FROM users WHERE user='$ID' AND pass='$Pass'
      だった場合、 SELECT * FROM users WHERE user='admin'--' AND pass='any'

ユーザーのID/PasswordをDBに保存するシステムを想定

SQL文の -- 以降は無視されるため、passwordの制約を無条件でクリアできる

SQLインジェクション 2/4

- sqli_1/readme.md を参考に、環境を構築してみましょう
  - 構築できたら、ブラウザで http://0.0.0.0:5000/ にアクセス
- ログインユーザーのデータベースには、現在adminしか登録されていません
  - adminユーザーとしてログインしてみましょう！
- ヒント
  - sqli_1/flask/hello.py を見ると、ログイン時のユーザー認証をどのようにしているかがわかります
  - SQL Injection Cheat Sheat なるものが沢山公開されています

SQLインジェクション 3/4

- おまけ
  - では、adminユーザーのパスワードそのものはわかるのでしょうか

- 今回の対策
  - 今回はsqlite3を直で利用 → flaskのプラグイン sqlalchemy を利用
    - クエリを直接書いてDBを呼び出すスタイルではなく、プログラム側にクエリ内容を分割して渡す
    - 特殊文字は sqlalchemy 側でサニタイズしてくれるので、SQL Injectionされにくい設計になる
  - sqlalchemy を使用したらSQL Injectionの事を考えなくても良いわけではなく、使い方次第でセキュリティホールを生んでしまうため、導入した上でコードの検証・テストが必要

SQLインジェクション 4/4

- 根本的対策
  - SQL文の組み立ては全てプレースホルダで実装する
  - SQL文の組み立てを文字列連結により行う場合は、エスケープ処理などを行うデータベースエンジンのAPIを用いる
  - Webアプリケーションに渡されるパラメータにSQL文を直接指定しない
- 保険的対策
  - エラーメッセージをそのままブラウザに表示しない
  - データベースアカウントに適切な権限を与える

出典：IPA 「安全なウェブサイトの作り方 改訂第7版」

おわりに

- 読み物・書籍紹介
  - IPA 「安全なウェブサイトの作り方」 (PDF)
  - 徳丸 浩 「安全なWebアプリケーションの作り方」(第2版 2018.06)
- 勉強会紹介
  - Security-JAWS (AWS)
  - IoTSecJP
- セキュリティ技術コンテスト紹介
  - CTF (Capture the Flag)

注意事項

不正アクセス禁止法違反に注意

今回のハンズオンで学んだことは、
インターネット上のサーバ
自分の管理外のサーバ
に対して絶対に実行しないで下さい！

参考

- IPA 「安全なウェブサイトの作り方」
- CTF for Girls #9 配布資料
- @IT Webアプリにおける11の脆弱性の常識と対策

演習の解法

XSS（クロスサイトスクリプティング）

- 自分の名前を入れて、打ち消し線で表示してみよう
  - `<s>なまえ</s>`
- フォントの色を変えてみよう
  - `<font color="red">なまえ</font>`
- 自分のcookieを表示してみよう
  - `<script>document.write(document.cookie);</script>`
  - Chrome の開発者ツールなどからも cookie は確認可能

ディレクトリトラバーサル

- このサイトの何処かに secret.txt があります。探してみましょう。
  - /image?image=../../../secret.txt
- このサイトのソースコード hello.py も落ちてます
  - /image?image=../../../web/hello/hello.py

SQLインジェクション

- adminユーザーとしてログインしてみましょう！
  - admin'–
- では、adminユーザーのパスワードそのものはわかるのでしょうか
  - python sqli_1/tools/blind_attack.py