セキュリティゼミ
成果報告

辻脇 優一
前田 泰晴

動機と目標

- 情報セキュリティという言葉は、一般的には、
  情報の機密性、完全性、可用性を確保することと定義される

  ✓ 機密性：認められた人だけが、ある情報にアクセスできる状態を確保すること
  ✓ 完全性：情報が破壊、改ざん又は消去されていない状態を確保すること
  ✓ 可用性：必要時に中断することなく、情報にアクセスできる状態を確保すること

  http://www.soumu.go.jp/main_sosiki/joho_tsusin/security/intro/security/index.html

- 情報セキュリティの機密性、完全性、可用性に関する
  攻撃手法と防御手法を学び、体験し、会社への適用を検討する

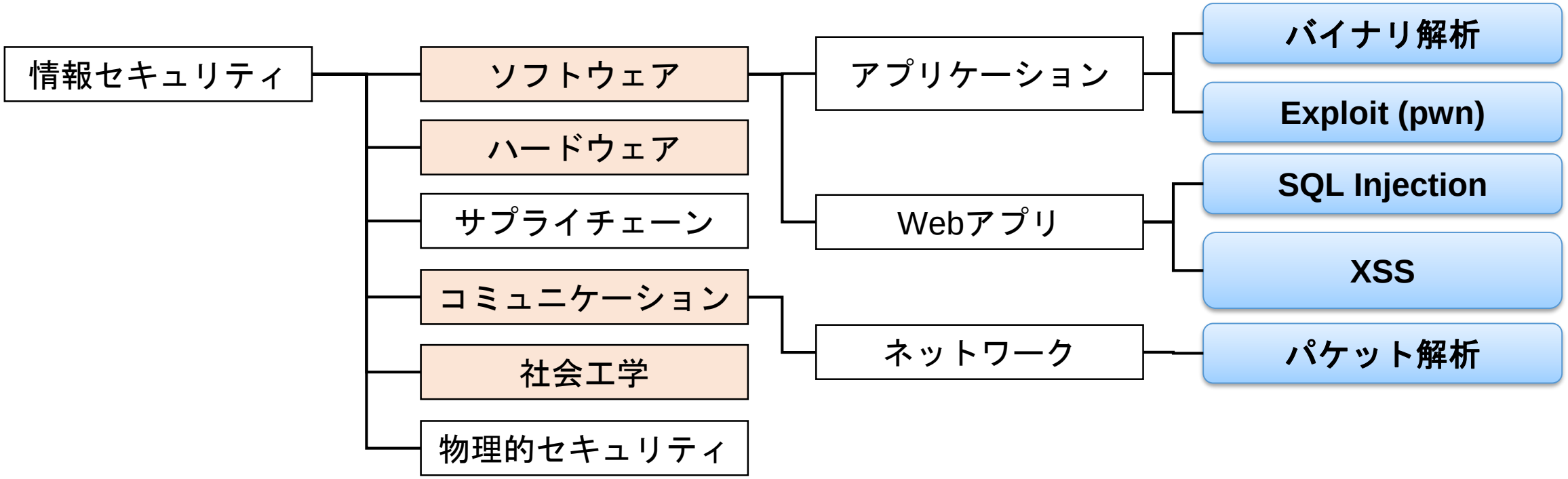
情報セキュリティ対策

- 利用者の立場
  - ✓ **ソフトウェアの更新**
  - ✓ **ウイルス対策ソフト（ウイルス対策サービス）の導入**
  - ✓ **IDとパスワードの適切な管理**

http://www.soumu.go.jp/main_sosiki/joho_tsusin/security/intro/beginner/index.html

- 開発者の立場
  - ✓ 上記に加えて、心がけるべきことがある

情報セキュリティ技術に関する調査対象



CAPECによる
ドメインの分類
(セキュリティー般)

SECCONによる
攻撃対象の分類

攻撃と解析の
手段の分類

各論

- バイナリ解析
- Exploit
- SQL Injection
- XSS (cross site scripting)

バイナリ解析

バイナリ　　：実行可能な形式を持ったデータファイル
バイナリ解析：バイナリの動作を様々な手法で把握すること

バイナリ解析のセキュリティにおける意義

- ✓ マルウェアの解析
- ✓ 脆弱性の診断

ただし、悪用も可能

- ✓ 脆弱性を発見し、攻撃の為に使う
- ✓ ソフト内の処理を盗む

バイナリ解析　一般的な手法

①表層解析

ファイルのフォーマット情報や
ファイル内文字列情報を取得

【使用ツール例】
・Linuxコマンド:
　file, readelf, stringなど

②動的解析

実際に実行する
解析する範囲を絞り込む

【使用ツール例】
・トレーサ:
　strace(システムコール)
　ltrace(ライブラリコール)
・デバッガ:
　GDB, OllyDbgなど

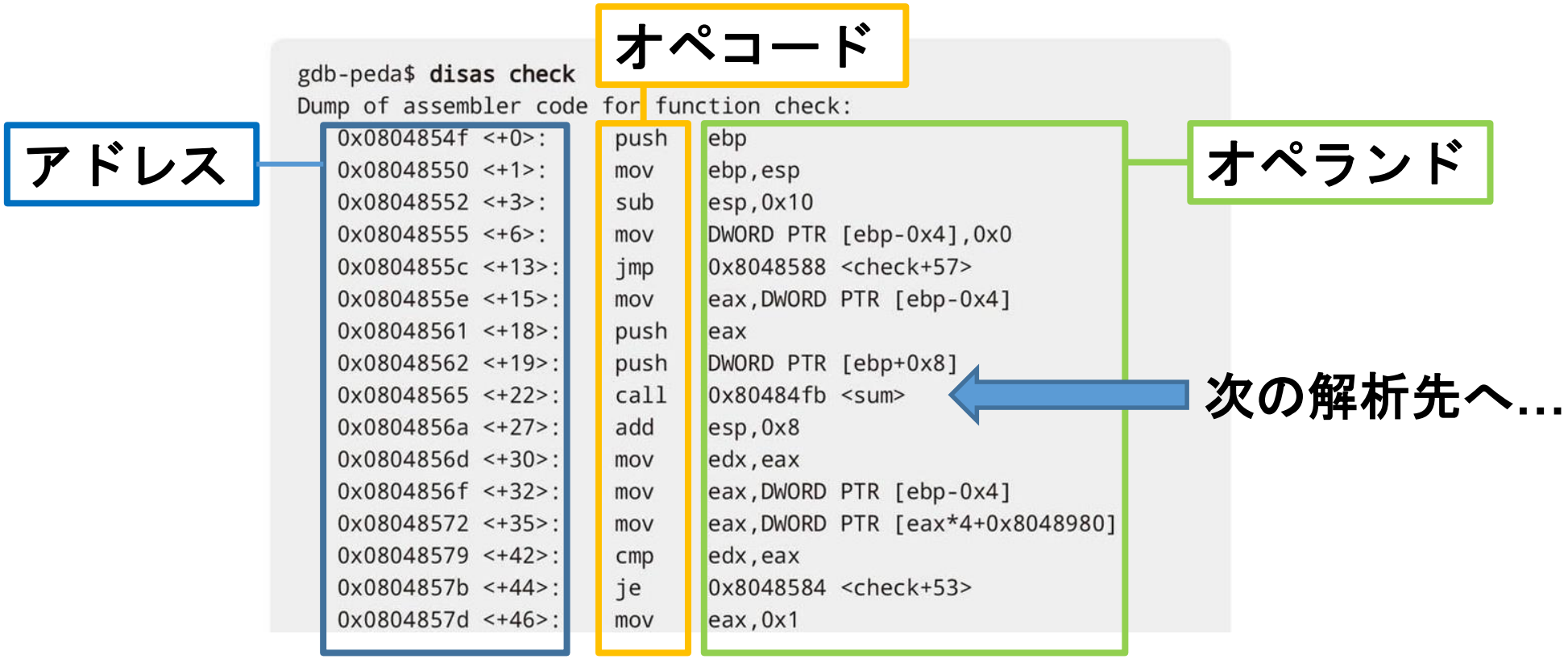
③静的解析

動的解析で絞った場所付近の
プログラムコードを逆アセンブル
非常に時間がかかる

【使用ツール例】
・逆アセンブラ:IDAなど

バイナリ解析　静的解析の例

あるチェック関数の逆アセンブル結果




```
gdb-peda$ disas check
Dump of assembler code for function check:
   0x0804854f <+0>:     push   ebp
   0x08048550 <+1>:     mov    ebp,esp
   0x08048552 <+3>:     sub    esp,0x10
   0x08048555 <+6>:     mov    DWORD PTR [ebp-0x4],0x0
   0x0804855c <+13>:    jmp    0x8048588 <check+57>
   0x0804855e <+15>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08048561 <+18>:    push   eax
   0x08048562 <+19>:    push   DWORD PTR [ebp+0x8]
   0x08048565 <+22>:    call   0x80484fb <sum>
   0x0804856a <+27>:    add    esp,0x8
   0x0804856d <+30>:    mov    edx,eax
   0x0804856f <+32>:    mov    eax,DWORD PTR [ebp-0x4]
   0x08048572 <+35>:    mov    eax,DWORD PTR [eax*4+0x8048980]
   0x08048579 <+42>:    cmp    edx,eax
   0x0804857b <+44>:    je     0x8048584 <check+53>
   0x0804857d <+46>:    mov    eax,0x1
```

バイナリ解析　対策

分析を遅延させるための様々な難読化の手法がある

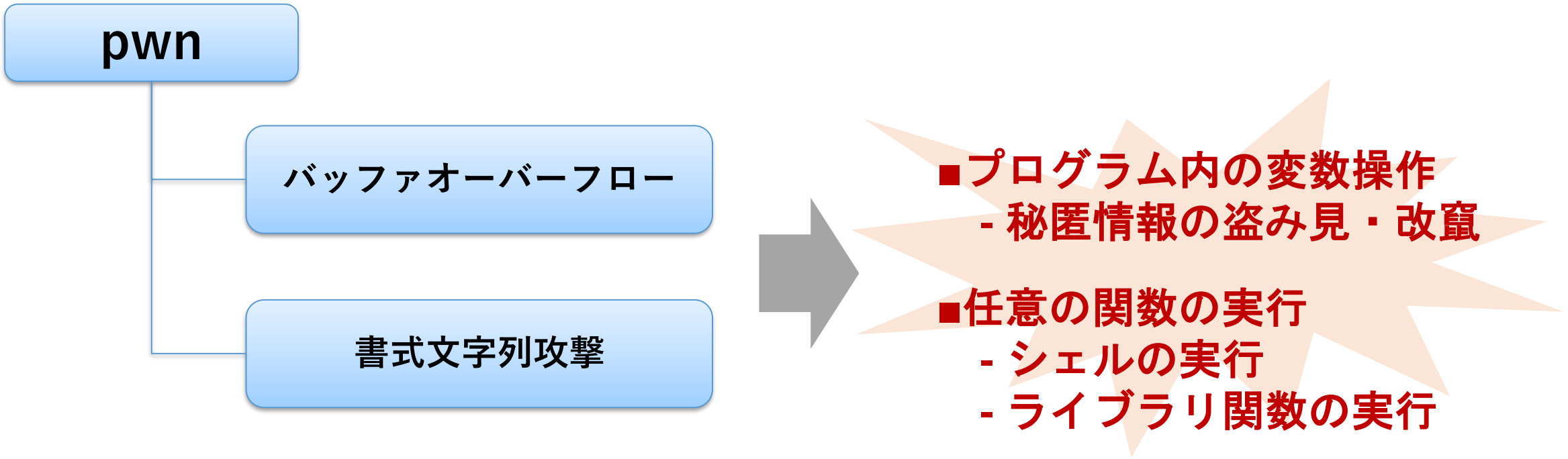
- ガベージコード
  本来の命令の間に意味のない命令を大量に配置
- コード置換
  簡単な命令を同等の動作をする複雑な命令セットに置換
- Anti-Disassembly
  常にFALSEの条件分岐を追加し、FALSE側に意味のない命令を大量に配置
- デバッガ検知
  プログラムが静的解析されているかどうかを検知し、妨害を行う

各論

- バイナリ解析
- Exploit
- SQL Injection
- XSS (cross site scripting)

Exploit

[概要] プログラムの脆弱性を利用して行う攻撃



Exploit　バッファオーバーフロー

[概要] スタック上に確保されたバッファ領域以上のサイズのデータを入力することで、変数・コールスタックの操作を行う攻撃

スタック内のデータを何に書き換えるかによって複数の分類がある

- バッファオーバーフロー
  - 特定変数
  - 特定アドレス
  - Return to PLT
  - Return to Libc
  - ROP

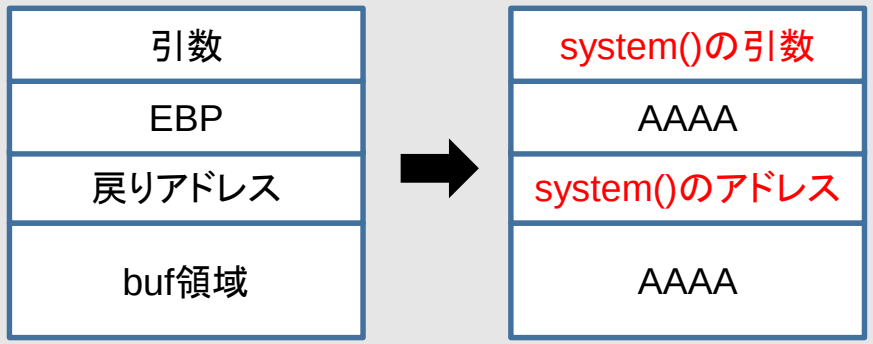
```
int main( int argc, char* argv[] )
{
    char buf[10];
    strncpy( buf, argv[1] );
    printf("%s¥n", buf );

    return 0;
}
```

argv[1]に10bytesを超えた以下の入力を与えると

AAA・・[system()のアドレス] AAAA [system()の引数]

スタックが以下のように破壊される



Exploit　書式文字列攻撃

[概要] printfやsyslogといった関数で表示する文字列に
書式文字列(%xや%nなど)を含ませることで行われる攻撃

```
int main()
{
   ＊＊＊その他のコード ＊＊＊

   char buf[256];
   fgets( buf , 256, stdin);

   printf( buf );

   ＊＊＊ その他のコード ＊＊＊

   return 0;
}
```

■ bufに%xを含ませた場合

%xはスタックから1つポップして16進数での出力を行う

➡ **任意の変数の盗み見ができる**

■ bufに%nを含ませた場合

%nは出力した文字数を特定の変数へ書き込みを行う

➡ **変数を任意の値に書き換えることができる**
関数のアドレス書き換えにより、任意の関数が呼べてしまう

Exploit　対策

① C, C++言語といった直接メモリを操作可能な言語を選択しない

② プログラミング時にユーザーからの入力文字列のチェックを行う
- ✓ バッファサイズが超えていないか
- ✓ 不正な文字列が入っていないか

③ コンパイラのセキュリティオプションを利用する
- ✓ スタック変更の検知(SSP)
- ✓ 不許可コード実行の防止(NX-bit)
- ✓ メモリのランダム化(ASLR)

各論

- バイナリ解析
- Exploit
- SQL Injection
- XSS

SQL Injection

SQLとは、データベース操作のための言語

SQL Injectionとはウェブの入力フォームなどに
「SQL文を含む文字列」を入力してDBへの操作を実行する攻撃

主な被害：データ窃取とデータ改ざん・削除

SQL Injection　簡単な例

```
<?php
  /*****  脆弱なコード例 *****/
  $username = $_POST[“username”];
  $password = $_POST[“password”];
  $query = “SELECT * FROM users WHERE username=‘{$username}’ AND password=‘{$password}”;
  $result = $db -> query($query) -> fetchAll();

  if($result)
      echo “ログイン成功”
  else
      echo “ログイン失敗”
```

Web上から入力したユーザ名とパスワードが…

正しければ…

ログイン成功！

ここでユーザ名に ‘ OR 1=1 -- と入力すると？

SELECT * FROM users WHERE username=‘’ OR 1=1 --’ AND password=‘{$password}”;

SQL Injection　対策

根本的対策

エスケープ処理の実施

特別な意味を持つ記号文字が普通の文字として解釈されるように処理する

保険的対策

エラーメッセージを非表示にする

- 詳細なデータベースに関するエラーメッセージをウェブページに表示させない
- エラーを表示するとしても、内容は最小限に

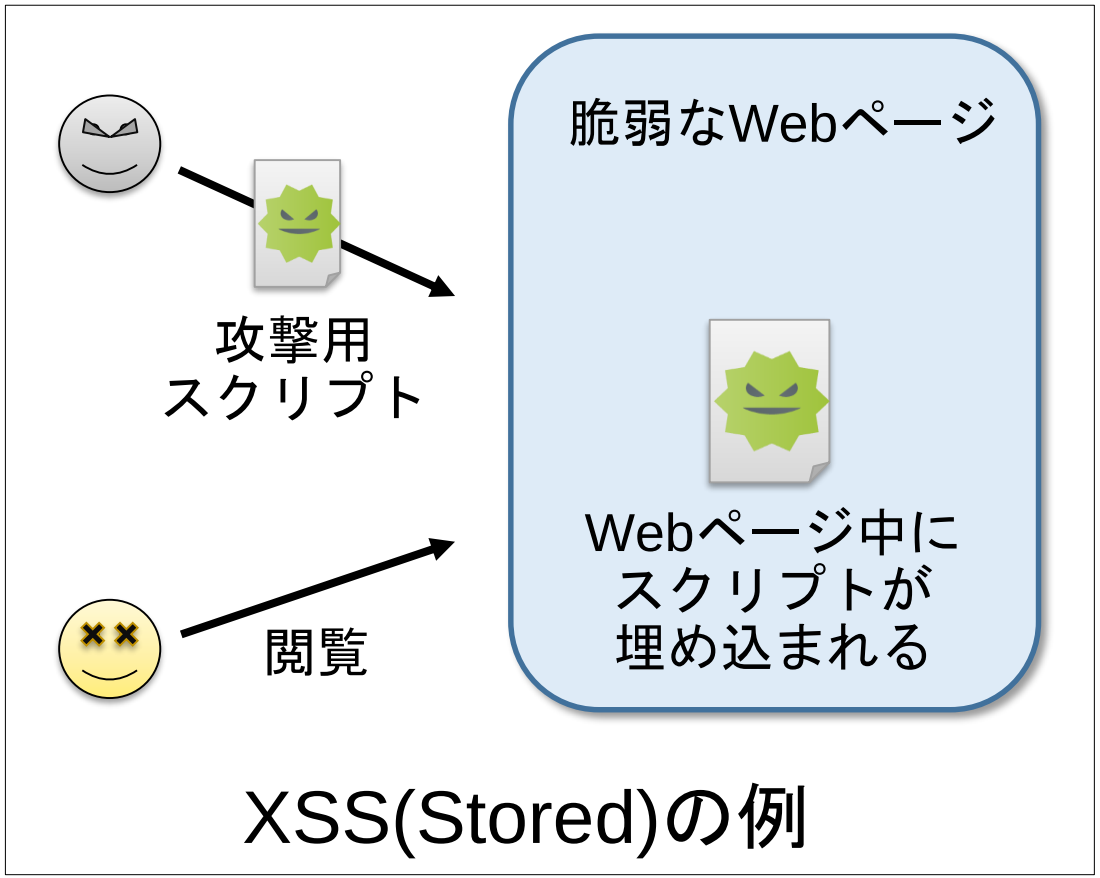
IPA: SQLインジェクション対策について
https://www.ipa.go.jp/files/000024396.pdf

各論

- バイナリ解析
- Exploit
- SQL Injection
- XSS

XSS

[概要]　Webサイト上に攻撃者のスクリプトを埋め込む攻撃



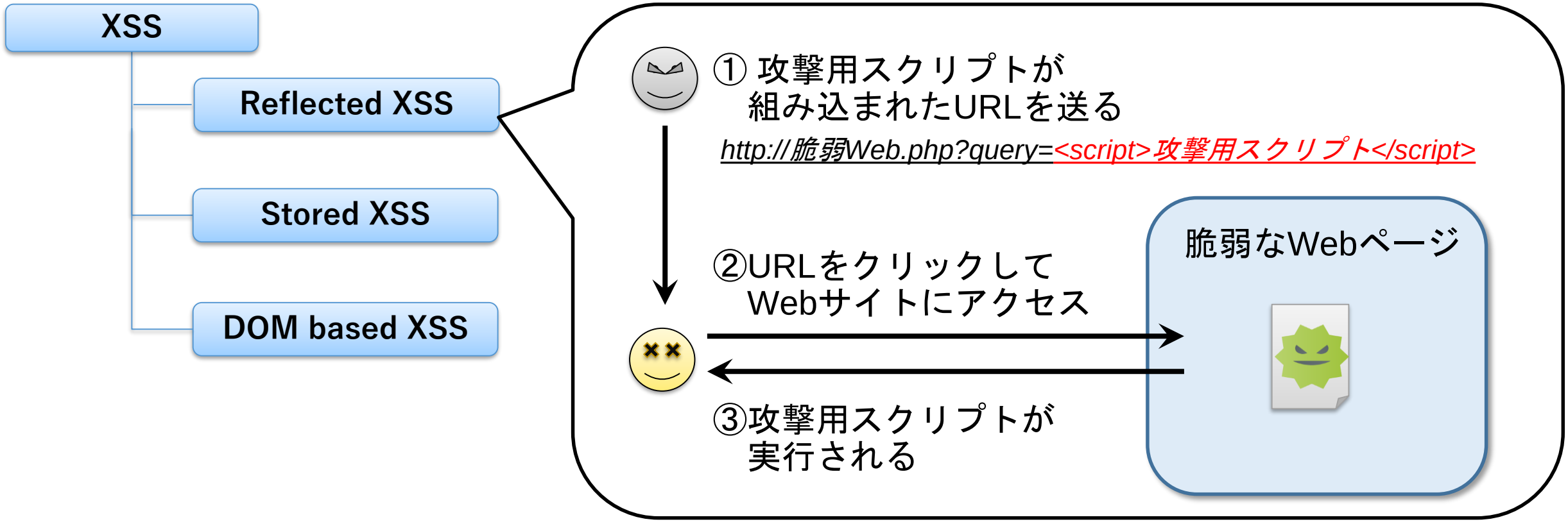
■Cookie情報の盗難

■ユーザー情報の不正送信

■Webページの改ざん

XSS　攻撃方法

XSSは3種類に分類される



XSS　対策

①セキュアコーディング
- Webページに出力する要素に対して、エスケープ処理を施す
- 入力されたテキストに対して構文解析を行い、スクリプトを除外する
- <script>タグを動的に生成しないコーディングにする

などなど

②Webページの設定
- HTTPレスポンスに文字コード設定
- 発行するCookieにHttpOnly属性を加える

などなど

③ XSS対策されたブラウザを使う

IPA 安全なウェブサイトの作り方
https://www.ipa.go.jp/security/vuln/websecurity.html

活動内容

1. 関連文献の調査
2. 「セキュリティコンテストのためのCTF問題集」の問題解決
3. SECCONへの参加

SECCONへの参加

情報セキュリティをテーマに多様な競技を開催する情報セキュリティコンテストイベントです。実践的情報セキュリティ人材の発掘・ 育成、技術の実践の場の提供を目的として設立されました。 https://2018.seccon.jp/seccon/about.html

■SECCON 2018参加概要(第1回)

[開催日] Web予選　2018年10月27日～10月28日 ※セキュリティゼミは10月から開始！！
[参加者] 653チーム
[問題数] 全25問
(Pwn:9　Reversing:6　Media:1　Web:1　QR:1　暗号:5　その他:2)

[結果]　正当数： 0 / 25 問

会社への適用(方針)

- 講習会等を通じて本ゼミで得た知識の社内での普及を図る
- 特に以下の対策に留意する
  - アプリケーション
    - 難読化ソフトを利用してバイナリ解析を困難にする
    - コンパイラのセキュリティ関連のオプションを利用する
  - Webアプリ
    - ユーザーからの入力をチェックする(サニタイズ)
  - ネットワーク
    - 暗号化して送る

参考文献

[1] **セキュリティコンテストのためのCTF問題集**
清水 祐太郎, 竹迫 良範, 新穂 隼人, 長谷川 千広, 廣田 一貴, 保要 隆明, 美濃 圭佑, 三村 聡志, 森田 浩平, 八木橋 優, 渡部 裕, SECCON実行委員会 (監修)
マイナビ出版 (2017/7/28)

[2] **セキュリティコンテストチャレンジブック -CTFで学ぼう! 情報を守るための戦い方**
碓井 利宣, 竹迫 良範, 廣田 一貴, 保要 隆明, 前田 優人, 美濃 圭佑, 三村 聡志, 八木橋 優, SECCON実行委員会 (監修)
マイナビ出版 (2015/9/30)

[3] **The Art of Unpacking**
Yason, Mark Vincent
Black Hat 2007

[4] 情報処理推進機構
https://www.ipa.go.jp/

[5] CAPEC セキュリティ攻撃パターン分類
https://capec.mitre.org/index.html