Tizenセキュリティ

Fourteenforty Research Institute, Inc.
株式会社 フォティーンフォティ技術研究所
http://www.fourteenforty.jp

シニア・リサーチ・エンジニア
鈴木 秀一郎

背景

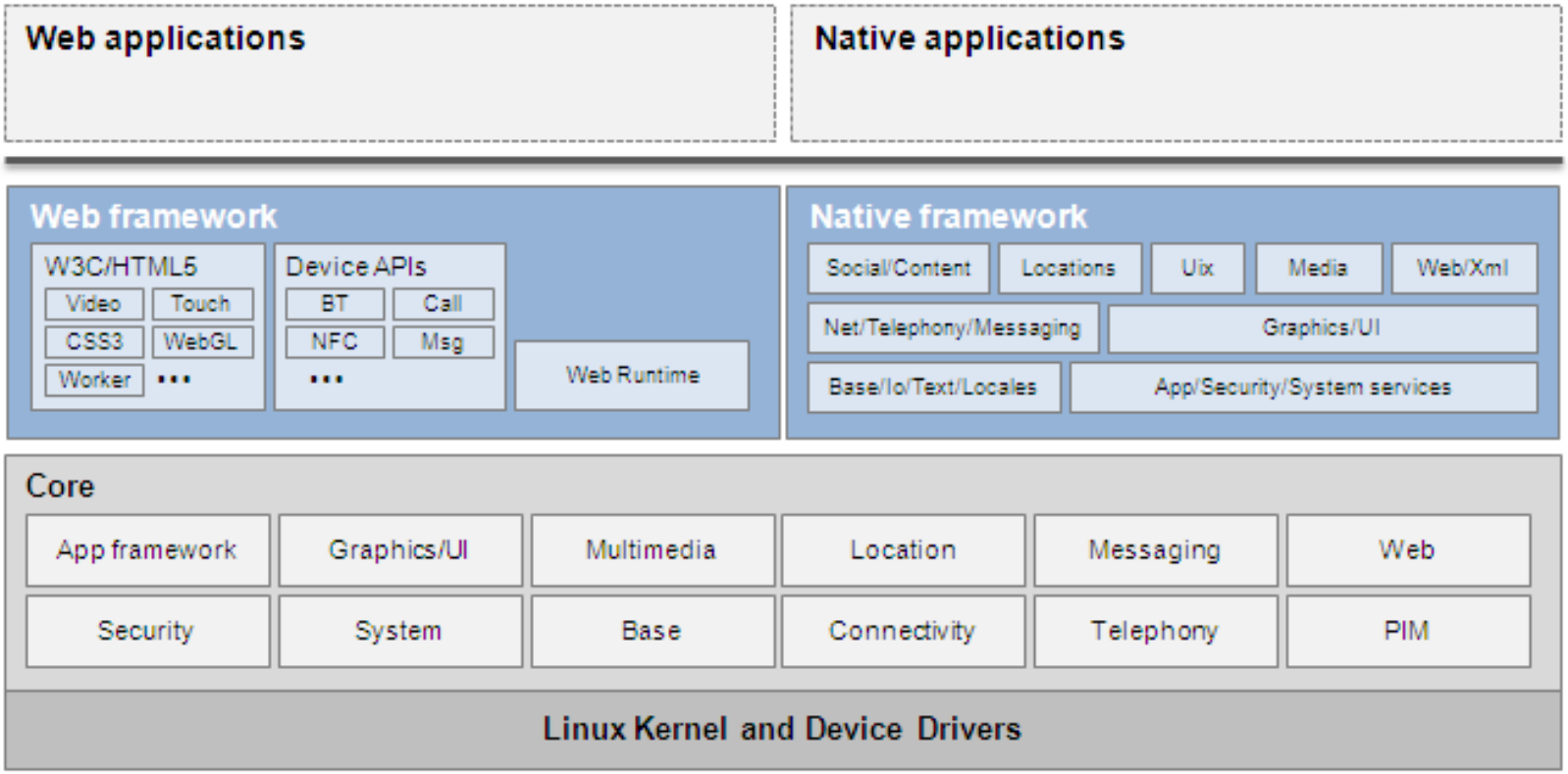
- iOS, Androidに代わる新たスマートフォン・タブレットOSとして、TizenやFirefox OSが注目されている
- 国内でも、年内か来年にはTizen OS搭載の端末がリリースされるとの情報
- Androidもリリース以降、常にセキュリティ面は注目されてきた
- 新たなOSのセキュリティはどうなのだろうか？
- 今回はTizenのセキュリティについて調査
- Tizen SDK 2.1のエミュレータを用いて調査

Tizen概要

- 概要
  - モバイルデバイス向けオープンソースOS
  - Linux Foundationのプロジェクトの一つ
  - サムスン電子とインテルによって主導されている
- 特徴
  - Linuxベース
  - Tizen 2.0からはWebおよびNative(C/C++)でのアプリ開発をサポート

Tizenアーキテクチャ

WebおよびNativeアプリをともにサポート



http://upload.wikimedia.org/wikipedia/commons/c/c3/What_is_tizen_architecture.pngより転載

Tizen Package

- 二つのパッケージフォーマット
  - Tizen Package (.tpk) : Native Applications
  - Tizen Web Package (.wgt) : Web Applications

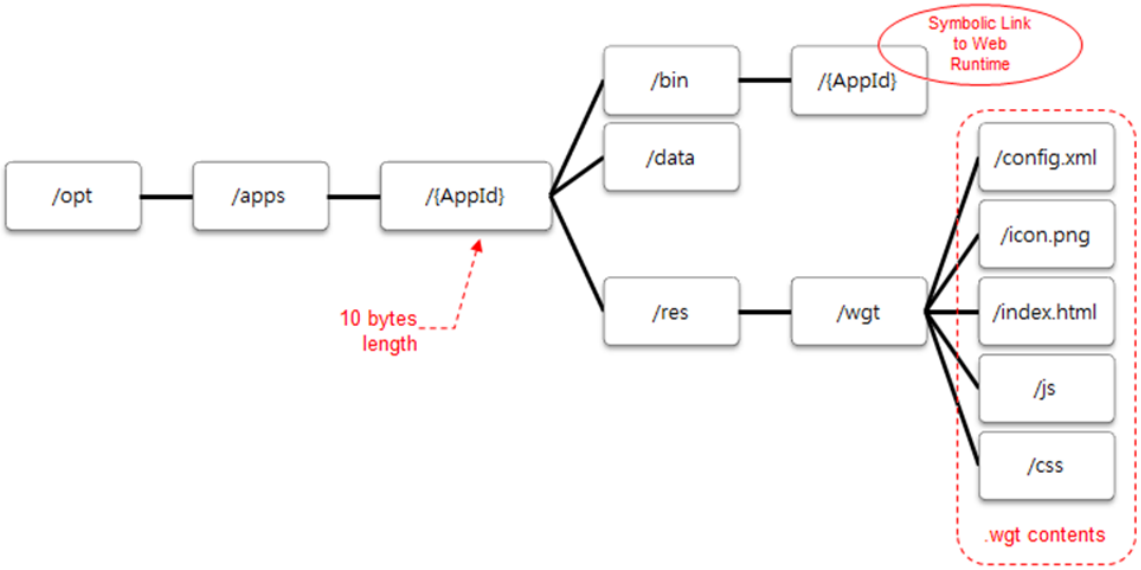
Tizen アプリ Web/Native 共通事項

- インストール先は /opt/apps/(AppId)
- AppIdは10バイトの文字列
- /opt/apps/(AppId)下は
  - bin/
  - data/
  - res/

  などが含まれる

Tizen Web package

- 拡張子 .wgt を持つZipアーカイブ
- .wgtファイルは以下の図の赤枠内の構造を持つ

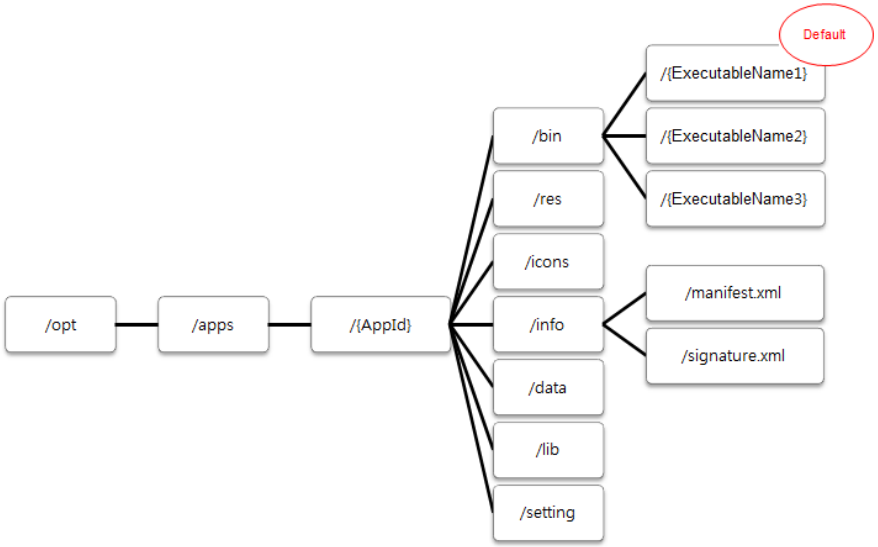


https://developer.tizen.org/documentation/articles/tizen-application-packaging-overviewより引用

- binディレクトリ下にWeb Runtime（WRT）へのシンボリックリンクが作成される
- 起動時にはこのファイルが実行されWebアプリをホストする

Tizen Native package

- 拡張子 .tpk を持つZipアーカイブ
- .tpkファイルはAppId下の構造をそのまま持つ



https://developer.tizen.org/documentation/articles/tizen-application-packaging-overviewより引用

- binディレクトリ下に直接Nativeコードのバイナリが配置される

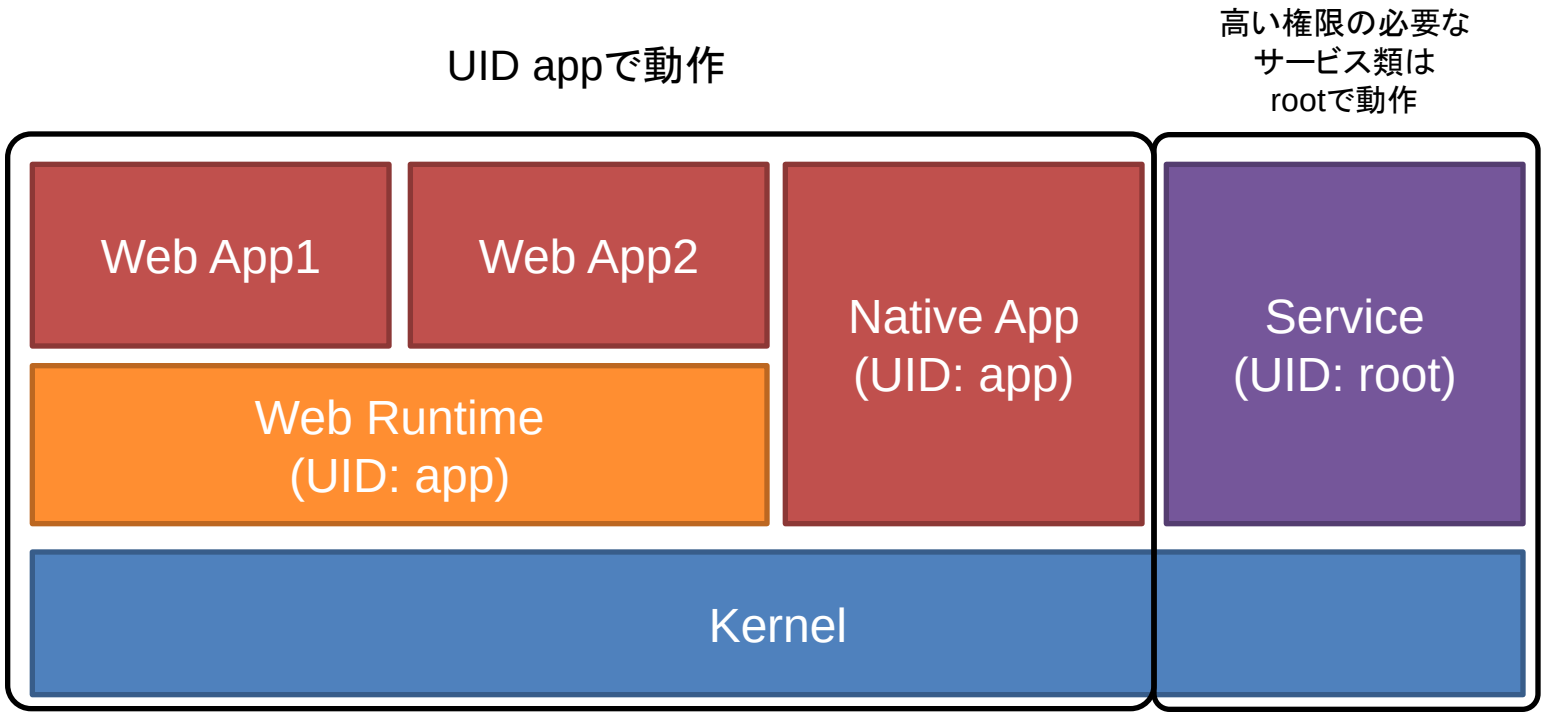
セキュリティ

- 以下の項目について、調査
  - OSレベルのセキュリティ
    - アクセス制御
    - 脆弱性攻撃防御
    - Content Security Framework
  - アプリの権限分離
    - Privileges
    - Feature

アクセス制御概要

- すべてのプロセスは以下の二つのユーザーで動作
  - root
  - app
- Webアプリ、NativeアプリともにUID “app”で動作
- SMACKによる強制アクセス制御
  - すべてのアプリはSmackでラベルづけされる

2つのUIDの利用


UID appで動作
高い権限の必要な
サービス類は
rootで動作
Web App1
Web App2
Native App
(UID: app)
Service
(UID: root)
Web Runtime
(UID: app)
Kernel

アプリ同士の分離

アプリはすべてUID “app”で動作

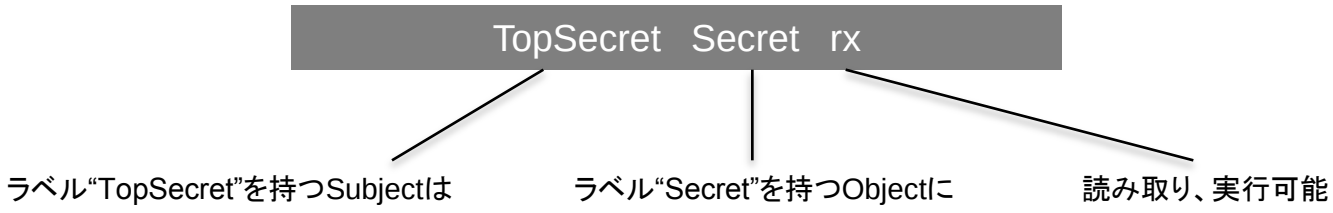
↓

アプリ同士は互いのファイルへアクセス可能？？

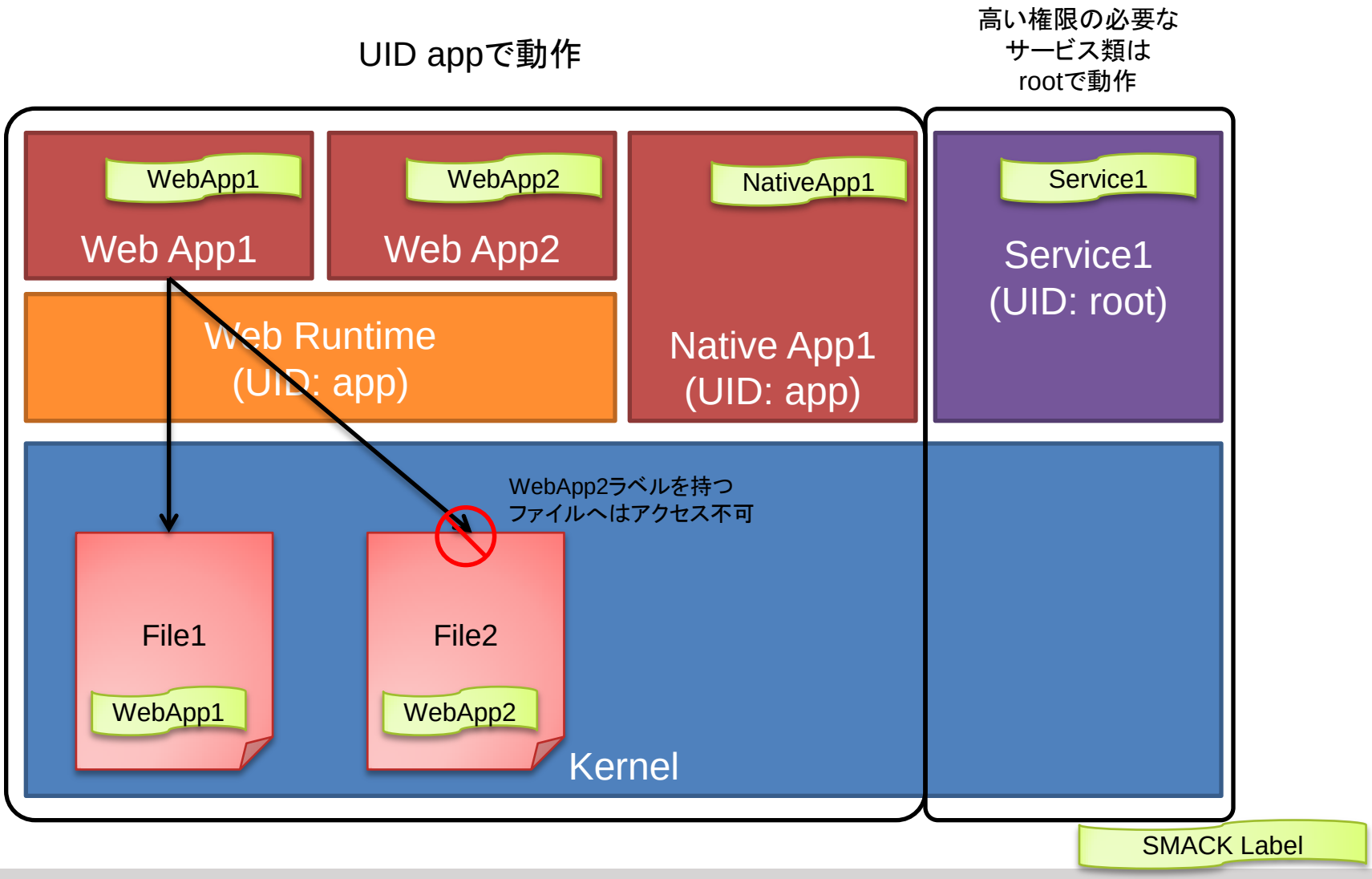
SMACKによるアクセス制御
（アプリ同士はデフォルトで干渉不可）

SMACK

- LSM(Linux Security Modules)の一つ
- Subject (≒プロセス)と Object (≒ファイル)にラベルを付け、ラベル間のアクセスルールを書くことでアクセス制御を行う
- アクセスルール例:



SMACK Labelの利用



脆弱性防御

- Tizen 2.0からNative (C/C++)によるアプリ開発をサポート
- 典型的なバッファオーバーフロー脆弱性の可能性
- ASLR および DEPが基本対策

DEP

・ 以下のコードをTizen Native Applicationとして実行

```
int func(){
    int a = 10;
    int b = 20;
    return a+b;
}

_EXPORT_ int OspMain(int argc, char *pArgv[])
{
        AppLog("Application started.");

        char buf[1024];
        int (*f)();
        memcpy( buf, (char*)func, 1024);
        f = (int (*)())buf;
        int b = f();

        ArrayList args(SingleObjectDeleter);
        args.Construct();
…..
```

スタックにバッファを用意

スタックにfuncをコピー

スタック上のコードを実行

特に問題なく動作
DEPは無効
（Tizen SDK 2.1 x86 Emulator上にて)

ASLR

- /proc/sys/kernel/randomize_va_space は 2
  ＝ ASLR有効であるという意味
- 実際には…（Tizen Native Appを用いてエミュレータで確認）
  - 同じプログラムを2回起動。/proc/[pid]/mapsの主要なモジュール、ヒープ、スタックのアドレスは2回とも結果は変化なし

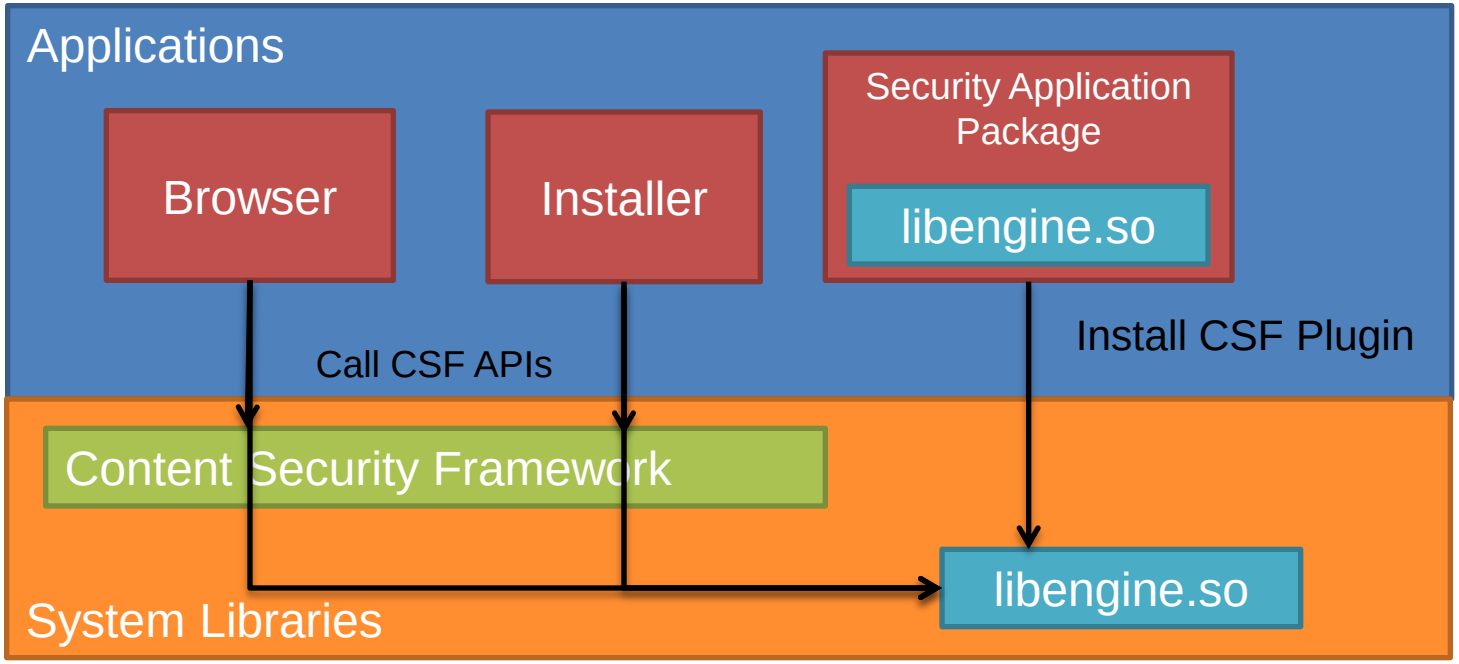
```
09e0e000-09e70000 rw-p 00000000 00:00 0          [heap]
09e70000-09f80000 rw-p 00000000 00:00 0          [heap]
b36e7000-b36ec000 r-xp 00000000 fe:00 73077      /opt/usr/apps/hNLQmS2Kl0/bin/MySample7.exe
b36ec000-b36ed000 rw-p 00004000 fe:00 73077      /opt/usr/apps/hNLQmS2Kl0/bin/MySample7.exe
b36ed000-b36f0000 r-xp 00000000 fe:00 73094      /opt/usr/apps/hNLQmS2Kl0/bin/MySample7
b36f0000-b36f1000 rw-p 00002000 fe:00 73094      /opt/usr/apps/hNLQmS2Kl0/bin/MySample7
bfdcf000-bfdf0000 rw-p 00000000 00:00 0          [stack]
```

→ ランダマイズされていない（Tizen SDK 2.1 x86 Emulator上にて）

- /proc/self/personalityの値が 00040000 (ADDR_NO_RANDOMIZE)
  - ASLRはこれによって無効化されている

Content Security Framework(CSF)

- Tizenにセキュリティチェック機能を柔軟に提供できるようにする仕組み
- CSFはAPIおよびPlug-inのインターフェースを規定
- セキュリティチェック機能はPlug-in (libengine.so)として提供
- Plug-inはファイル、URL、Webサイト(HTML, JavaScript)などのチェック機能を提供
- Plug-inを提供するアプリケーション(Security Application Package)には信用された署名が必要

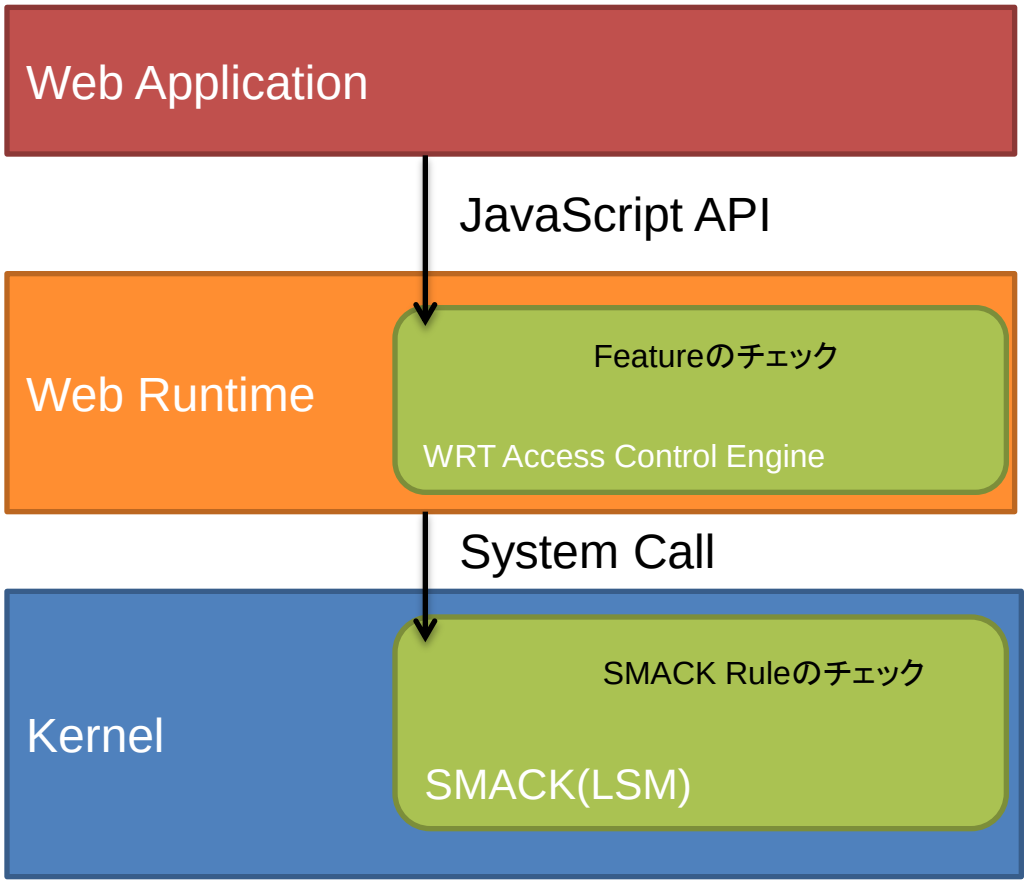


アプリの権限分離

- Privilege
  - TizenではAPIを3つの権限の違いで分けている
    - Public　－すべての開発者が利用可能
    - Partner　－Tizen storeのパートナー登録者のみが利用可能
    - Platform　－Tizenプラットフォームの管理用（一部の開発者のみ利用可能）
  - 適切なPrivilegeを持たないアプリはそのAPIを呼び出せない
  - Privilegeはアプリのマニフェストファイルに記述
- Feature
  - Androidのパーミッションに相当する仕組み
  - マニフェストファイルに利用機能（連絡先へのアクセス、カメラへのアクセスなど）を記述
  - Web Runtimeは内部にAccess Control Engine(ACE)を保持
  - ACEが各機能へのアクセス制限を行う

Webアプリケーションサンドボックス

Web RuntimeおよびSMACKによる2段階のアクセス制御



仮にWeb Runtimeに脆弱性があっても、System Call時に許可のないデバイス・ファイルへのアクセスを防げる

（すべての場合についてWRTとSMACK Ruleのアクセス制御が同じレベルになるかは未調査）

まとめ

- SMACKによるアクセス制御がセキュリティの中核
- Webアプリについては、WRTおよびSMACKによる2段階のアクセス制御
- Content Security Frameworkを提供することで、柔軟にセキュリティチェック機能を提供可能
- ネイティブコードによるアプリ開発を許可していることでバッファオーバーフローなど古典的な脆弱性の可能性
- ASLR/DEPの今後改良の余地が残る

参考資料

- http://download.tizen.org/misc/media/conference2012/tuesday/ballroom-c/2012-05-08-1600-1640-tizen_security_framework_overview.pdf
- http://download.tizen.org/misc/media/conference2012/wednesday/seacliff/2012-05-09-0945-1025-understanding_the_permission_and_access_control_model_for_tizen_application_sandboxing.pdf
- http://www.youtube.com/watch?v=GtiAQOo4beg