JPCERT CC®

C/C++ セキュアコーディング

File I/O part2:
ファイルシステムに関する脆弱性

2010年3月23日

JPCERTコーディネーションセンター

File I/O part2 :ファイルシステムの脆弱性

JPCERT CC®

Part 1

パーミッションと権限

- ユーザ識別所有者権限パーミッション
- プロセスの権限
- 権限の変更と管理
- パーミッションの管理
- 脅威の緩和方法

Part 2

ファイルシステムの脆弱性

- **パストラバーサルの脆弱性、ファイルパス、シンボリックリンクとハードリンク**
- **脅威の緩和方法**
- **chroot(), jail()**

ここ

Part 3

Race Condition

- 並列処理
- 競合状態・排他制御・デッドロック
- ロックファイルとファイルロック
- ファイルシステムへの攻撃
- 脅威の緩和方法

UNIX POSIX

Copyright® 2010 JPCERT/CC All rights reserved.

本モジュールのゴール

JPCERT CC®

- ✓ ファイルシステムに関して脆弱性が作り込まれるポイントを把握する

  どのようなコードに？ 何が問題で？

- ✓ 脆弱性を作り込まないための脅威緩和アプローチを習得する

Copyright® 2010 JPCERT/CC All rights reserved.

もくじ

JPCERT CC®

ファイルシステムに関する脆弱性
- ディレクトリトラバーサル
- 等値エラー
- シンボリックリンク
- 正規化
- ハードリンク
- 特殊ファイル

脅威の緩和アプローチ

まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

デモ

Copyright® 2009 JPCERT/CC All rights reserved.

もくじ

JPCERT CC®

ファイルシステムに関する脆弱性

- ディレクトリトラバーサル
- 等値エラー
- シンボリックリンク
- 正規化
- ハードリンク
- 特殊ファイル

脅威の緩和アプローチ

まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

パスの等値を利用した脆弱性

JPCERT CC®

<概要>

攻撃者がアプリケーションのアクセスコントロール(無害化メカニズム)を攻略する手法のひとつ

- リソースに「異なる」が「同等」の名前を与える

正規化エラーの一種

<想定される影響>

ファイルシステムの情報漏えい

Copyright® 2010 JPCERT/CC All rights reserved.

等値エラーの種類

1. シングルドットのディレクトリ: /./
2. 大文字と小文字の区別
3. フォーク
4. その他(末尾文字など)

Copyright® 2010 JPCERT/CC All rights reserved.

1. シングルドットのディレクトリ: /./

– EServ の脆弱性。パスワード保護ファイルにアクセスできる。

  – サーバ上の保護されたディレクトリadminの内容にアクセスできるウェブリクエストを作成可能

– 攻撃者が細工したリクエスト

  http://host/./admin/

– プログラムは以下と等値として処理してしまう

http://host/admin/

Etype Eserv Directory Traversal Vulnerability
http://www.securityfocus.com/bid/773

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

Macintosh Hierarchical File System (HFS+)は大文字と小文字を区別しない

/home/PRIVATE == /home/private

一方 Apache はケースセンシティブ

/home/PRIVATE != /home/private

Copyright® 2010 JPCERT/CC All rights reserved.

2. 大文字と小文字の区別$_2$

OS X 上のApache の設定

```
<Directory /Library/WebServer/Documents/test>
          Order deny,allow        Deny from all
</Directory>
```

GET /test/index.html は 403 Forbidden で通らないが…

GET /TeSt/index.html は通してしまう！

パスの等値を検証できていないことが脆弱性につながった

MacOS X Client Apache File Protection Bypass Vulnerability

http://www.securityfocus.com/bid/2852/

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

HFS/HFS+のデータフォークとリソースフォーク

– データフォーク: ファイルの実データを格納

sample.txt/..namedfork/data は **sample.txt** と同じ

Apple HFS+ ファイルシステム上で実行される Apache の脆弱性

– ファイル名ではなくデータフォークでアクセスすれば Apache file handle をバイパス可能

Ref: CVE-2004-1084

ファイルシステムが異なるフォークをサポートする場合、アプリケーションはそれを想定して正しく動作すべき

Copyright® 2010 JPCERT/CC All rights reserved.

4．パスの等値エラーの他の例

‘filename.’ （末尾のドット）

‘file.name’ （途中のドット）

‘filename ‘ （末尾のスペース）

‘ filename’ （先頭のスペース）

‘file name’ （途中のスペース）

‘//filename’ （先頭のダブルスラッシュ）

‘filename/’ （末尾のスラッシュ）

すべて’filename’と等しいと評価される問題

正規化で解決できる問題

CWE-41: Improper Resolution of Path Equivalence

http://cwe.mitre.org/data/definitions/41.html

Copyright® 2010 JPCERT/CC All rights reserved.

もくじ

JPCERT CC®

ファイルシステムに関する脆弱性
- ディレクトリトラバーサル
- 等値エラー
- シンボリックリンク
- 正規化
- ハードリンク
- 特殊ファイル

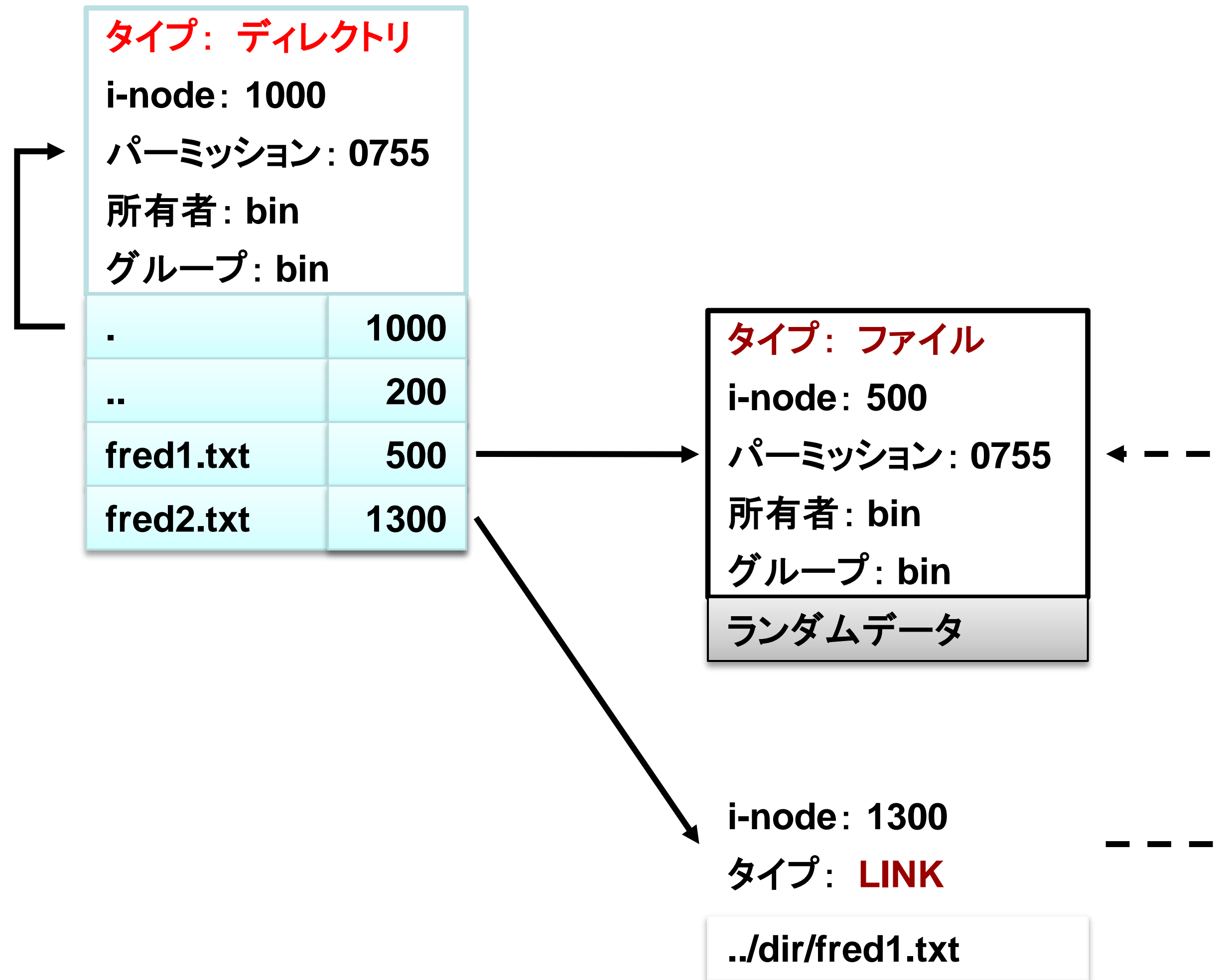
脅威の緩和アプローチ

まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

シンボリックリンク

JPCERT CC®



Copyright® 2010 JPCERT/CC All rights reserved.

シンボリックリンクの脆弱性の例

仮定

– root 権限を持つ setuid root プログラムとして実行される
– 攻撃者は write()呼び出しで書き込まれる userbuf のデータを制御できる

```
fd = open("/home/kubo/.conf", O_RDWR);
if (fd < 0) abort();
write(fd, userbuf, userlen);
```

このコードのどこが脆弱？

Copyright® 2010 JPCERT/CC All rights reserved.

単純な攻撃シナリオ

JPCERT CC®

1. 攻撃者は .conf から /etc/passwd へのシンボリックリンクを作成

```
% cd  /home/kubo
% ln -s /etc/passwd .conf
```

2. 脆弱なプログラムを実行
   - root として書き込むためにファイルをオープン
   - 攻撃者が制御する情報をパスワードファイルに書き込む

```
% runprog
```

3. たとえばパスワードなしの root アカウントを新規作成することが可能
4. 攻撃者は次に su コマンドを使って root アカウントに切り替え、ルートにアクセス

```
% su
#
```

Copyright® 2010 JPCERT/CC All rights reserved.

シンボリックリンクを操作する関数

JPCERT CC®

シンボリックリンクが参照するファイルではなく、シンボリックリンクファイル自体に対して動作する関数

- **unlink()** シンボリックリンクファイルを削除
- **lstat()** シンボリックリンクファイルに関する情報を返す
- **lchown()** シンボリックリンクファイルのユーザとグループを変更
- **readlink()** 指定されたシンボリックリンクファイルの内容を読み取る
- **rename()** 引数 old から new へシンボリックリンクの名前を変更したり、new が存在する場合そのシンボリックリンクファイルを上書きする

シンボリックリンクをたどる操作かどうかを意識してコーディングしよう

Copyright® 2010 JPCERT/CC All rights reserved.

シンボリックリンクの特徴

JPCERT CC®

- 存在しないファイルへのリンクを作成できる
- シンボリックリンクが指すファイル名が変更されたり、ファイルが移動・削除されたりしても、シンボリックリンクは存在し続ける
- 任意のファイルへのリンクを作成できる
- ファイルシステム上 readable でないファイルへのリンクも作成できる
- パーティションやディスク境界をまたがって存在するファイル同士をリンクできる
- シンボリックリンクを変更することで、使用中のアプリケーションのバージョンを変更したり、公開するウェブサイト全体をごっそり変更できることも

Copyright® 2010 JPCERT/CC All rights reserved.

シンボリックリンクの脆弱性の成立条件

JPCERT CC®

安全

- セキュアなディレクトリの中
  - /home/me （通常ユーザのホームディレクトリは、初期設定で安全なパーミッションを設定されている）

危険

- /tmp などの共有ディレクトリ内での操作
  - 他のユーザのディレクトリで管理者など昇格した権限で行う操作（アンチウイルスプログラムをadmin権限で実行するなど）

正規化を用いてシンボリックリンクの脆弱性を回避する

Copyright® 2010 JPCERT/CC All rights reserved.

もくじ

JPCERT CC®

ファイルシステムに関する脆弱性

- ディレクトリトラバーサル
- 等値エラー
- シンボリックリンク
- 正規化
- ハードリンク
- 特殊ファイル

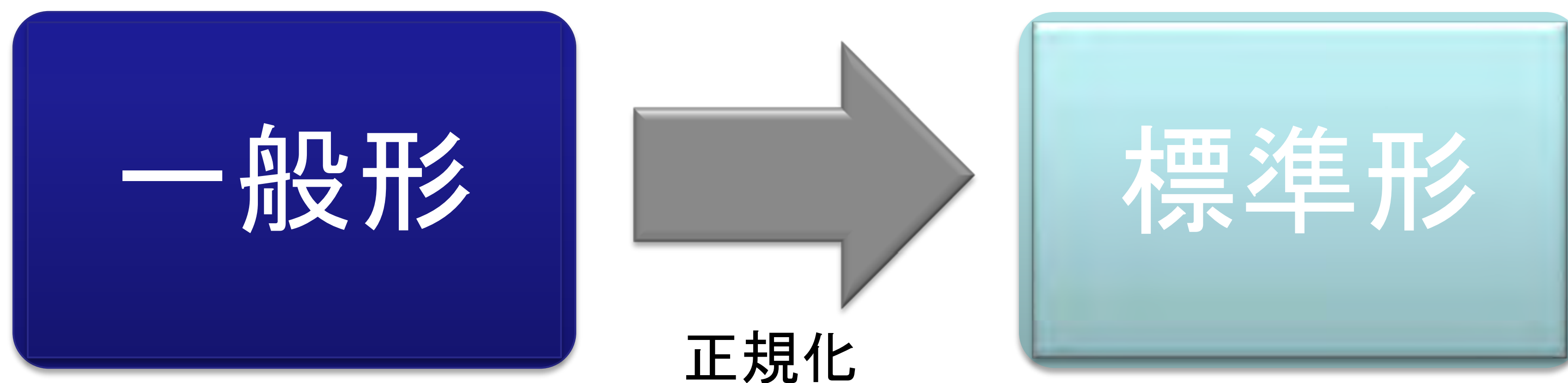
脅威の緩和アプローチ

まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

正規化(canonicalization)とは

- ファイル名やパスを標準形(絶対パス)にすること



Copyright® 2010 JPCERT/CC All rights reserved.

正規化の目的

JPCERT CC®

解決したい問題

- パス名、ディレクトリ名、ファイル名には、検証を困難・不正確にする文字が含まれる可能性がある
- パス名のなかにシンボリックリンクが含まれていると、ファイルの実体やファイルの同一性が分かりにくい

解決方法

⇒ファイル名を標準形に変換。検証を容易化。

- OSやファイルシステム間で異なる標準形は、OS固有の正規化メカニズムを用いるのがベスト

FIO02-C. *Canonicalize path names originating from untrusted sources*

Copyright® 2010 JPCERT/CC All rights reserved.

標準形(Canonical Form)

JPCERT CC®

- 正規化で解決できる問題
  - パストラバーサル
  - 同値エラー
  - シンボリックリンク問題
- 解決できない問題
  - ハードリンク
  - 特殊ファイル
- 標準形はシンボリックリンクを含んではいけない。
  - /usr/../home/robert は /home/robert と同値
  - /home/robert 正規化されたパスである

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

以下のシンボリックリンクが存在する。

`/home/alfred/sss -> /home/myhomebiz/accounting/spreadsheets/`

Q. 問題

`/home/bob/../mary/../alfred/.//sss/may.xls`
を正規化したパスは次のうちどれか？

a） `/home/alfred/sss/may.xls`

b） `/home/myhomebiz/accounting/spreadsheets/may.xls`

c） `/home/alfred/may.xls`

Copyright® 2010 JPCERT/CC All rights reserved.

UNIXにおけるパスの正規化

JPCERT CC®

POSIX の **realpath()** 関数はファイルを正規化して絶対パスを返す。

- シンボリックリンクを解決
- 余分な '/' や
- /./ や /../ なども解決する

注： GNU libc4、libc5、および BSD の **realpath()** 実装にはバッファオーバーフローの脆弱性があった。 libc-5.4.13 で修正されている。(10 年以上も潜在していた) [VU#743092]

この脆弱性が修正されたrealpath関数を使う！

Copyright® 2010 JPCERT/CC All rights reserved.

改訂版 realpath() の実装

POSIX.1.2008 で **realpath()** が改訂

– 最近の glibc や Linux 実装では、 **resolved_name** に NULL ポインタが渡すと、関数内でメモリを割り当てた上で解決された名前を格納する
– 旧版(POSIX.1)の **realpath()** はNULLポインタが渡されたときの動作は処理系依存

ただし、**realpath()**が改訂版かどうか検出する可搬性のある方法は存在しない…

Copyright® 2010 JPCERT/CC All rights reserved.

改訂版 realpath()の使用例

```
#include <stdlib.h>
...
char *symlinkpath = "/tmp/symlink/file";
char *actualpath;
actualpath = realpath(symlinkpath, NULL);
if (actualpath != NULL){
    ... use actualpath ...
    free(actualpath);
}else{
    ... handle error ...
}
```

解決するファイルパス

```
#include <stdlib.h>
char *realpath(const char *, char *restrict);
```

正規化したパスを返す

正規化したパスを保存する場所

Copyright® 2010 JPCERT/CC All rights reserved.

旧版 realpath() と PATH_MAX

JPCERT CC®

旧版の**realpath()**は、 **resolved_name** が正規化したパスを十分格納できる文字配列を参照していると想定して動作

- バッファのサイズが **PATH_MAX** あれば十分だが、**PATH_MAX** がマクロとして定義される保証はない…

PATH_MAX が定義されていれば、**PATH_MAX** の大きさを持つバッファを割り当て、**realpath()** の結果を保持

Copyright® 2010 JPCERT/CC All rights reserved.

PATH_MAX を使った旧版 realpath() の使用例

JPCERT CC®

```
char *canonical_file = NULL;
canonical_file = malloc(PATH_MAX);
if(realpath(argv[1], canonical_file) == NULL) {
    /* エラー処理 */
}
/* ファイル名を確認 */
fopen(canonical_file, "w");
/* ... */
free(canonical_file);
```

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

GCC ユーザは次の GNU 拡張も使える。

```
#define _GNU_SOURCE
#include <stdlib.h>
char *
canonicalize_file_name(const char *)
```

realpath(path, NULL)
と等価

Copyright® 2010 JPCERT/CC All rights reserved.

Windows における正規化は複雑

Windows にはファイルの命名方法が多数あるため、正規化はさらに複雑

- 汎用命名規則 (UNC) 共有
- ドライブマッピング
- 短い名前(8.3 形式)
- 長い名前
- Unicode 名
- 特殊ファイル
- 末尾のドット、フォワードスラッシュ、バックスラッシュ
- ショートカット
- …

Copyright® 2010 JPCERT/CC All rights reserved.

もくじ

JPCERT CC®

ファイルシステムに関する脆弱性

- ディレクトリトラバーサル
- 等値エラー
- シンボリックリンク
- 正規化
- ハードリンク
- 特殊ファイル

脅威の緩和アプローチ

まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

ハードリンクの特徴

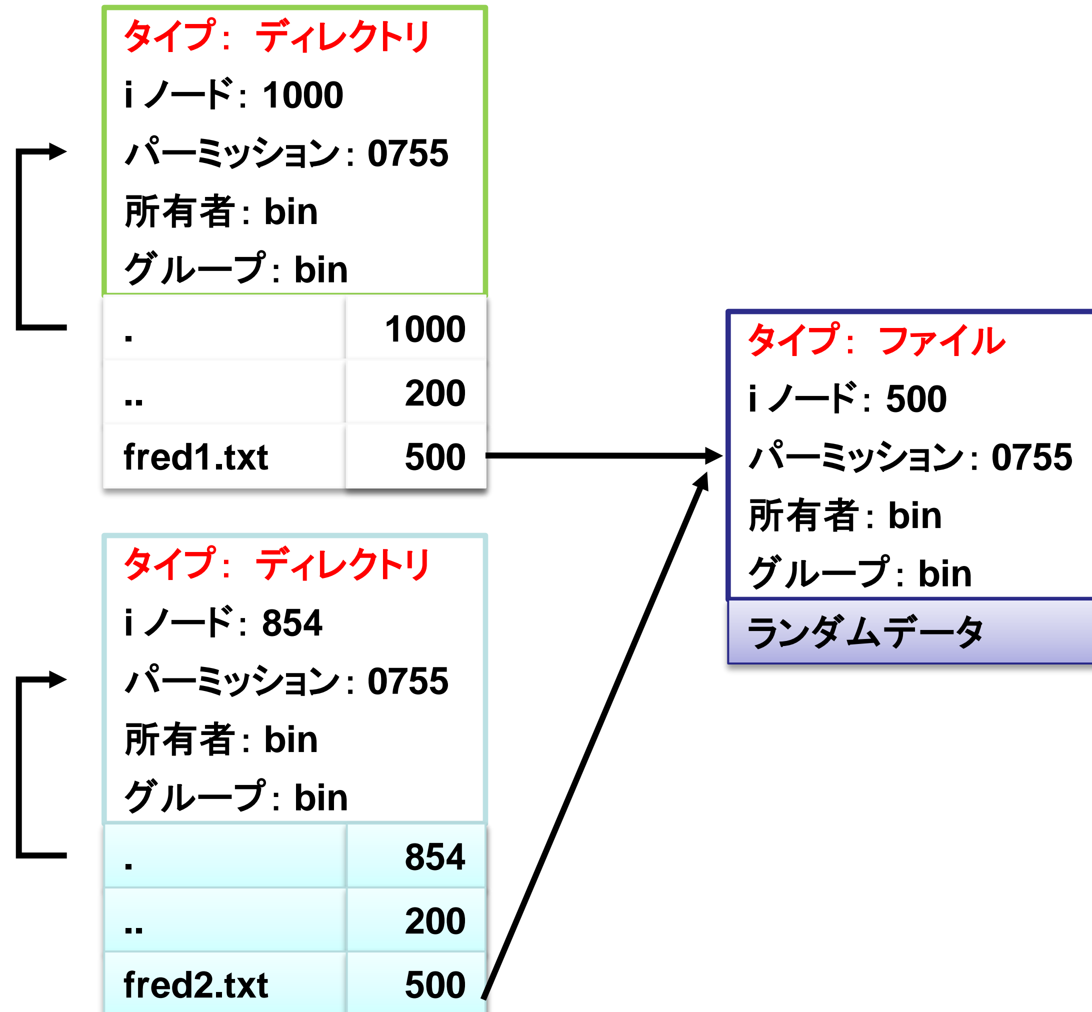
JPCERT CC®

- `ln` コマンドで作成。たとえば `ln /etc/passwd` は以下を行う。
  - inode 内で、`passwd` ファイルのリンクカウンタをインクリメント
  - カレントディレクトリ内に新しいディレクトリエントリを作成
- 元のディレクトリエントリと区別できない
- ディレクトリを参照したり、ファイルシステムをまたがることはできない
- 所有権とパーミッションは inode に属すため、同じ inode のすべてのハードリンクは同じ所有権とパーミッションを持つ
- ファイルへのすべての参照が削除されない限り、ひとつのハードリンクを削除してもファイルは削除されない
  - 参照は、ハードリンクか open ファイル記述子のいずれか
  - リンクカウンタが 0 の場合のみ inode を削除できる
  - すべてのハードリンクが削除されない限り、元の所有者はディスククオータを解放できない

Copyright® 2010 JPCERT/CC All rights reserved.

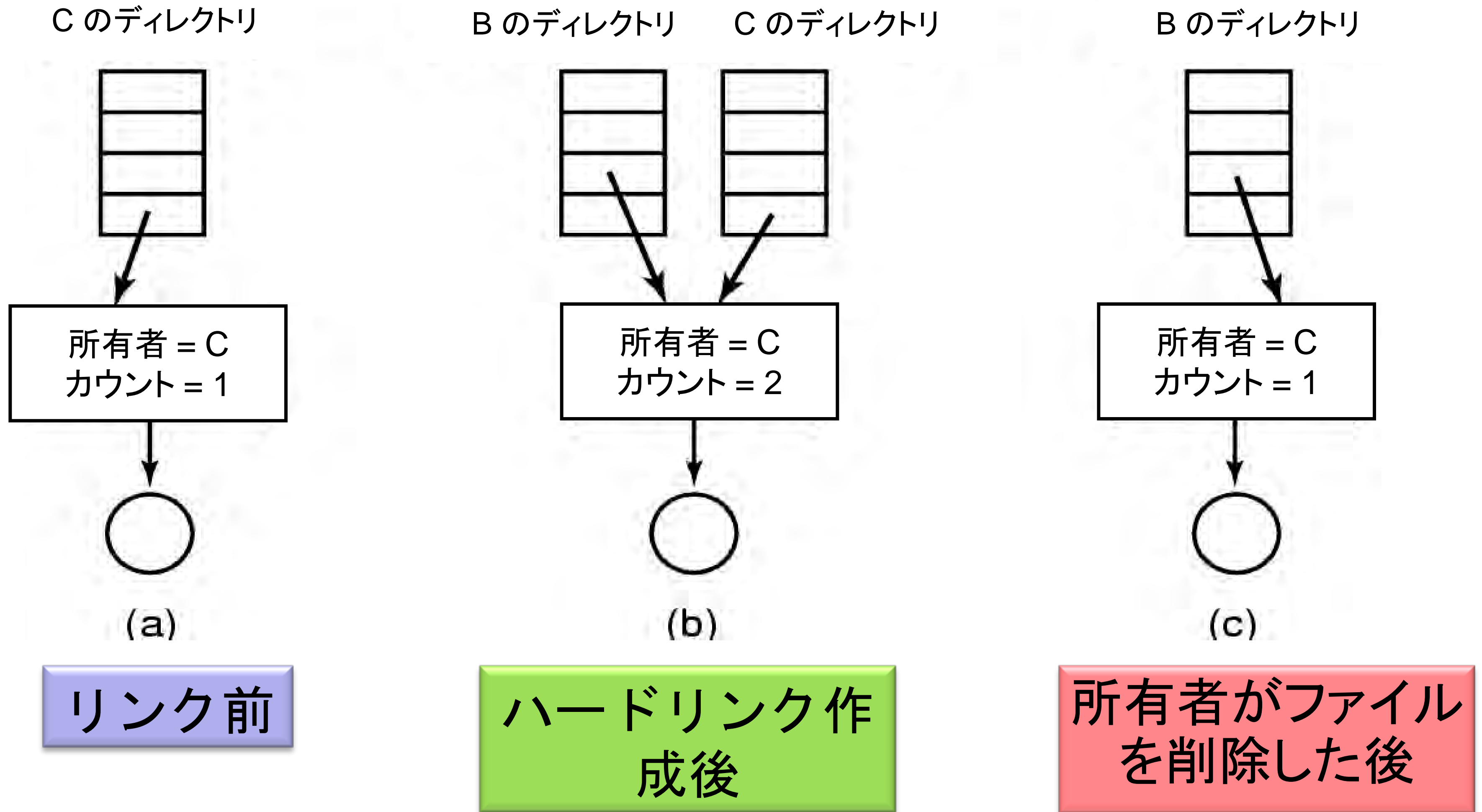
ハードリンクの例

JPCERT CC®



Copyright® 2010 JPCERT/CC All rights reserved.

ハードリンクで共有されるファイルと inode



Copyright® 2010 JPCERT/CC All rights reserved.

ハードリンク脆弱性の例

JPCERT CC®

次のコードが setuid root プログラムとして実行されるとどんな問題がある？

```
stat stbl;
if (lstat(fname, &stbl) != 0)
  /* エラー処理 */
if (!S_ISREG(stbl.st_mode))
  /* エラー処理 */
fd = open(fname, O_RDONLY);
```

lstat() はシンボリックリンクの参照先ではなく、シンボリックリンク自身を stat する

シンボリックリンクならこのテストはパスしない

ハードリンクを補足できない。攻撃者は、fname に /etc/passwd へのハードリンクを参照させることができる。

– fname がハードリンクするすべてのファイルの内容を読み取れる

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

リンクカウントが1でなければ、ファイルへのパスが複数存在することが分かる。

```
struct stat stbl;
if ( (lstat(fname, &stbl) == 0) && // ファイルが存在
     (!S_ISREG(stbl.st_mode)) && // 通常ファイル
     (stbl.st_nlink <= 1) ) { // ハードリンクなし
  fd = open(fname, O_RDONLY); // オープンしても問題ない！
}
else {
  /* エラー処理 */
}
```

このコードには競合状態も存在します。詳しくは、次のモジュールで解説します

Copyright® 2010 JPCERT/CC All rights reserved.

別パーティションに機密ファイルやユーザファイルを保存する

- パーティションをまたぐハードリンクは作成できない
  - ハードリンク攻撃防止策（/etc/passwd へのリンクなど）
- システム管理的にもよりセキュア
- ただし、プログラムの実行環境がそうであるとは限らない

Copyright® 2010 JPCERT/CC All rights reserved.

ハードリンク vs. シンボリックリンク

JPCERT CC®

ハードリンク	シンボリックリンク
リンクされるファイルと inode を共有	独自のファイル(独自の inodeを持つ)
リンクされるファイルと同じ所有者とパーミッション	リンクされるファイルとは別の所有者とパーミッション
常に存在するファイルにリンク	存在しないファイルを参照可能
ファイルシステムをまたがったりディレクトリに対して動作しない	ファイルシステムをまたがったりディレクトリに対して動作する
ノードへの元のリンクと新しいリンクとの区別がつかない	シンボリックリンクと他のファイルタイプを容易に区別できる

Copyright® 2010 JPCERT/CC All rights reserved.

もくじ

ファイルシステムに関する脆弱性

- ディレクトリトラバーサル
- 等値エラー
- シンボリックリンク
- 正規化
- ハードリンク
- 特殊ファイル

脅威の緩和アプローチ

まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

ディレクトリ

- drwxr-xr-x /

シンボリックリンク

- ファイルパスのテキスト表現 （別のファイルへの参照）
- lrwxrwxrwx termcap -> /usr/share/misc/termcap

名前付きパイプはプロセス間通信に利用され、ファイルシステムのどこにでも存在できる。

- mkfifo コマンドで作成：mkfifo mypipe
- パーミッションフィールドの頭に p がつく

signal-based attack
に悪用されることも！

prw-rw---- mypipe

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

UNIXドメインソケット

- 同一のマシン上で実行中の2つのプロセス間通信を許可
- パーミッションフィールドの先頭に **s** がつく

```
srwxrwxrwx X0
```

デバイスファイル

- 適切なデバイスドライバにアクセス権を適用し、デバイスを操作するために利用される
  - キャラクタデバイス：シリアルストリームの入出力のみを提供（先頭に **c** **がつく**）
  - ブロックデバイス：ランダムにアクセス可能（先頭に **b** がつく）

```
crw------- /dev/kbd

brw-rw---- /dev/hda
```

Copyright® 2010 JPCERT/CC All rights reserved.

Linux デバイス名

Linux では、ファイルではなくデバイスを開くことで特定のアプリケーションをロックできることがある。

- /dev/mouse
- /dev/console
- /dev/tty0
- /dev/zero

ウェブブラウザに file:///dev/mouse を開かせるとマウスがロック。リブートしないと元にもどらない Linux の問題。

- 攻撃者が <IMG SRC=file:///dev/mouse> のようなイメージタグを埋め込んだサイトをつくったら…

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

stat() 関数と S_ISREG() マクロを使って通常ファイルを特定する。

```
struct stat s;
if (stat(filename, &s) == 0) {
  if (S_ISREG(s.st_mode)) {
    /* ファイルは通常ファイル */
  }
}
```

Copyright® 2010 JPCERT/CC All rights reserved.

もくじ

JPCERT CC®

ファイルシステムに関する脆弱性

- ディレクトリトラバーサル
- 等値エラー
- シンボリックリンク
- 正規化
- ハードリンク
- 特殊ファイル

脅威の緩和アプローチ

まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

外部からオブジェクトの直接参照ができない設計

例）

ID 1を “inbox.txt” にマッピング

ID 2 を “profile.txt” にマッピング

アーキテクチャ・設計レイヤ

Copyright® 2010 JPCERT/CC All rights reserved.

サンドボックスでプログラムを実行

JPCERT CC®

- "jail"などサンドボックス環境でプログラムを実行。プロセスとOSの境界を明確にする。
  - Unix **chroot()** や AppArmor.
- ただし、サンドボックスはOSへの脅威を緩和してくれるだけ。プログラム自身のその他の箇所は保護してくれない。
- **chroot()**は CWE-243など仕様上の注意を良く守って使う。

CWE-243: Failure to Change Working Directory in chroot Jail
http://cwe.mitre.org/data/definitions/243.html

アーキテクチャ・設計レイヤ

Copyright® 2010 JPCERT/CC All rights reserved.

クライアント/サーバの両方でチェックを二重化

JPCERT CC®

クライアント/サーバモデルのアプリの場合、クライアント側でのセキュリティチェックに依存しない。

CWE-602: Client-Side Enforcement of Server-Side Security
http://cwe.mitre.org/data/definitions/602.html

攻撃者がクライアント側のチェックをバイパスすることを想定した設計にする。

アーキテクチャ・設計レイヤ

Copyright® 2010 JPCERT/CC All rights reserved.

全ての入力を疑う

- ホワイトリスト方式を使う
  - “accept known good” アプローチ
- 仕様に合致しない入力はすべて拒否、あるいは合致した形式に変更する
- ブラックリスト方式を使い、想定外の入力を拒否する

実装レイヤ

Copyright® 2010 JPCERT/CC All rights reserved.

パスは正規化する

JPCERT CC®

`realpath()`などパスを正規化する関数を用い、". ."などを取り除く

以下の脆弱性を防ぐ

- **CWE-23: Relative Path Traversal**
  - http://cwe.mitre.org/data/definitions/23.html
- **CWE-59: Improper Link Resolution Before File Access ('Link Following')**
  - http://cwe.mitre.org/data/definitions/59.html

実装レイヤ

Copyright® 2010 JPCERT/CC All rights reserved.

最小権限の原則を守る

JPCERT CC®

OSの権限管理メカニズムを使う

権限を持たないユーザで実行し、あらゆる攻撃の影響を小さくする

プログラムのインストール

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

- 静的解析ツールを使い、この手の脆弱性を検出
- ファジングなど動的検査も有効
- 手作業によるコードレビューや侵入検査、脅威分析も有効

Copyright® 2010 JPCERT/CC All rights reserved.

プロセスのファイルシステムへのアクセスを制御するしくみ。

- chroot()
- jail()

Copyright® 2010 JPCERT/CC All rights reserved.

JPCERT CC®

プロセスのルートディレクトリを変更

- プロセスがアクセスできるシステムリソースを制限
  - システムリソースへの無許可のアクセスを防止
  - 新しいツリー内で“..”, symlink、その他の攻撃から保護
- 呼び出しにスーパーユーザ権限が必要

```
chdir /tmp/ghostview
chroot
/tmp/ghostview
su tmpuser
```

潜在的問題

- カレントディレクトリは変更してくれない
  - chdir() をし忘れると、プロセスは変更されたルートから抜け出せる
- UID を変更し忘れるとプロセスが抜け出す可能性

Copyright® 2010 JPCERT/CC All rights reserved.

jail()

FreeBSD で最初に導入された

chroot() ＋ ネットワークも制限

- 各 jail は1つの IP アドレスしか持たず、ジェイル内のプロセスは他の IP アドレスを使って通信できない
- 同じジェイル内の他のプロセスとだけやり取りする

使用上の注意点

- 制限されたディレクトリにプログラムが必要なユーティリティーが全てあるとは限らない
- ユーティリティをコピーすると攻撃者に危険な武器を与えることになるかも
- ネットワーク通信は制御できない

Copyright® 2010 JPCERT/CC All rights reserved.

サンドボックスが有効なコード

```
enum { array_max = 100 };

/* 特権で動作。argv[1] と argv[2] はユーザ入力 */

char x[array_max];
FILE *fp = fopen(argv[1], "w");
strncpy(x, argv[2], array_max);
x[array_max - 1] = '¥0';

if (fwrite(x, sizeof(x[0]), sizeof(x)/sizeof(x[0]), fp) <
    sizeof(x)/sizeof(x[0])) {
  /* エラー処理 */
}
```

Copyright® 2010 JPCERT/CC All rights reserved.

FIO16-C. ジェイルを作成してファイルのアクセス制御を行う

JPCERT CC®

```
/* ディレクトリ chroot/jail はカレントディレクトリにあること。またディレクトリに適切なパーミションを設定し、全
   てのファイルシステムディスクリプタをクローズする。*/

if (setuid(0) == -1) { /* エラー処理 */ }
if (chroot("chroot/jail") == -1) { /* エラー処理 */}
if (chdir("/") == -1){ /* エラー処理 */}

/* 権限を永久に破棄 */
if (setgid(getgid()) == -1) { /* エラー処理 */}
if (setuid(getuid()) == -1) { /* エラー処理 */}

enum {array_max = 100};
FILE *fp = fopen(argv[1], "w");
char x[array_max];
strncpy(x, argv[2], array_max);
x[array_max - 1] = '¥0';

/* jail の中での書き込みは安全 */
if (fwrite(x, sizeof(x[0]), sizeof(x)/sizeof(x[0]), fp) <
    sizeof(x)/sizeof(x[0])) {
  /* エラー処理 */
}
```

Copyright® 2010 JPCERT/CC All rights reserved.

もくじ

JPCERT CC®

- ファイルシステムに関する脆弱性
  - ディレクトリトラバーサル
  - 等値エラー
  - シンボリックリンク
  - 正規化
  - ハードリンク
  - 特殊ファイル
- 脅威の緩和アプローチ
- まとめ

Copyright® 2010 JPCERT/CC All rights reserved.

脆弱性と緩和方法のまとめ

JPCERT CC®

脆弱性	緩和方法
ディレクトリトラバーサル	正規化
等値エラー	正規化
シンボリックリンク	正規化
ハードリンク	リンクカウントのチェック パーティションの分離
特殊ファイル	ファイルタイプのチェック

Copyright® 2010 JPCERT/CC All rights reserved.

まとめ

JPCERT CC®

- ユーザインタフェースや他の外部 API を経由し、ファイルシステムのディレクトリ構造やファイル名を公開しない

- パス名、ディレクトリ名、ファイル名に基いて判断しない。ファイル名とファイルの実体の間には、緩やかな相互関係しかない。

- OS 固有の正規化の方法を使う

- 特定のファイルシステムを前提にコーディングしない

Copyright® 2010 JPCERT/CC All rights reserved.

参考資料

[Meunier 04] Pascal Meunier. CS390S: Canonicalization and Directory Traversal, November 2004.

[MITRE 07] MITRE. Common Weakness Enumeration, Draft 7. October 2007. http://cwe.mitre.org

[Howard 02] Howard, Michael & LeBlanc, David C. *Writing Secure Code*, 2nd ed. Redmond, WA: Microsoft Press, 2002 (ISBN 0-7356-1722-8).

[Viega 03] Viega, John & Messier, Matt. *Secure Programming Cookbook for C and C++: Recipes for Cryptography, Authentication, Networking, Input Validation & More*. Sebastopol, CA: O'Reilly, 2003 (ISBN 0-596-00394-3).

CERT C/C++ セキュアコーディングスタンダード

Input Output (FIO)のセクション

https://www.securecoding.cert.org

Copyright® 2010 JPCERT/CC All rights reserved.