# Secure by Default

## OpenBSD と河豚板がお送りする

## 「他にはない安全・安心」

川俣吉広
fuguita.org

# ドラフト版です

- ターゲット層： OSC or NISOC あたり
- ガチの BSD ユーザ向けではない
  - 上記イベント、

    あるいは Qiita/Zenn のボリューム層あたり
- 現況：構成案と完成版の中間くらい

# 世間を騒がしたニュース

- 2026 年 4 月

  **Assessing Claude Mythos Preview’s cybersecurity capabilities**

  https://red.anthropic.com/2026/mythos-preview/

  - **A 27-year-old OpenBSD bug**

    → TCP SACK の符号付き整数オーバーフローによりカーネルクラッシュ

    → OpenBSD で **27 年間**発見されなかった脆弱性を **AI** が発見

    → OpenBSD 7.8 patch-025 にて修正済

**“OpenBSD—an operating system known primarily for its security”**

**「セキュリティの高さで知られるオペレーティングシステムである OpenBSD 」**

*の脆弱性を AI が発見 !!*

ということで様々なメディア・ SNS で取り上げられる

ところで

**OpenBSD の**

**「セキュリティが高い」って具体的にはどういうこと？**

→今回のお題

# OpenBSD とは

- BSD(Berkeley Software Distribution) 由来の OS
  →Linux とは別物

- Linux との違い (BSD 系 OS の特徴 )
  - カーネルだけはなく OS 全体をプロジェクトで一括して開発
    - ライブラリ、コマンド、マニュアルなど全部 ...

# プロジェクト目標

- portability（移植性）
- standardization（標準への準拠）
- correctness（正確性）
- proactive security（先制的なセキュリティ）
- integrated cryptography（統合された暗号機能）

→ セキュリティだけがプロジェクト目標ではない

# OpenBSD のセキュリティ（設計・運用）

- Proactive security（先制的なセキュリティ）
  → 問題が起こる前に、事前に問題点を潰しておく
- 継続的・網羅的なソースコードの監査
- 定期的なリリース、迅速な Errata 対応
- 過去のしがらみを引き継がない
- シンプルな実装、適切なデフォルトの選択
- 決定論的なセキュリティ機能 ≠ オプション

# ソースコードの監査

- input はどこから来るのか
- 最大長は保証されているか
- 呼び出し元でチェックされているか
- buf の大きさは本当に十分か
- 別の経路からこの関数が呼ばれた場合はどうか
- エラー時にどうなるか
- 文字列が NUL 終端されている保証はあるか
- その関数が root 権限で動いていたら何が起こるか
- etc...

```
char buf[100];
strcpy(buf, input);
```

シンプルなソースコード

# ソースコードの監査(2)

- 「脆弱性の発見」が目的ではない
- まず「プログラムとして正しく動くか」という視点
- セキュリティホールとバグを区別しない
  - 「正しく動かない部分を修正する」という姿勢
  - →その結果、後でセキュリティホールが塞がれていたことも
- 一たび見つかった不具合は、システム全体にわたり見直し
- man page の不備も「バグ」

# 定期的なリリース・迅速な対応

- CVS にてソースコードを管理
  - Main trunk = -CURRENT ブランチ
    - 半年に一回リリースブランチ ( 例： OPENBSD_7_9) を分岐
      → 安定版として Errata を適用しながら 1 年間メンテ
  - Errata( 修正情報 ) の発行
    - ソースレベルでの修正箇所・修正理由の開示
    - i386/amd64/arm64 に対してはバイナリパッチを提供
      →syspacth コマンドで適用

# 過去のしがらみを引きずらない

- 独自実装
  - OpenSSH
  - IPF → PF
  - OpenSSL → LibreSSL
  - Apache → Nginx
    → OpenBSD httpd
  - Sendmail → OpenSMTPD
  - ntpd.org → OpenNTPD

  など多数

- libc, libcrypto など
  ライブラリの頻繁な bump

# OpenBSD のセキュリティ（技術面）

## “OpenBSD Innovations”

https://www.openbsd.org/innovations.html（約 150 件）

セキュリティに関する概念・機能・プログラムやサブシステムなどの一覧

## 多重防御

- **攻撃面を減らす** – 少ない / 適切なデフォルト設定
- **攻撃成功を困難に** – RETGUARD, W^X, KARL/ASLR など
- **攻撃成功での影響限定** – 特権 { 分離 , 放棄 }, pledge/unveil

# 攻撃面を減らす(1)

“Only two remote holes in the default install, in a heck of a long time!”

- インストール直後、外部に開いているのは SSH のポートのみ
- SSH のデフォルト設定は
  「PermitRootLogin prohibit-passwd」
  (root ログインはパスワード認証禁止)

# 攻撃面を減らす(2)

- 「OpenBSD は、**必要ないサービスは有効にしない**。web サーバにしたければ、ユーザはそれなりに調べて勉強して、自分でそれを有効にしなきゃいけない。だからユーザは、**結果として自分のシステムでなにが動いているか把握できる**。でも、初心者がすぐに使えるわけではない。勉強しなきゃいけない。**セキュリティのためには、知識のあるユーザ**というのも（いやそれこそが）**とてもだいじだ**」

OpenBSD の巣窟 （山形浩生，2000 頃）
https://cruel.org/openbsd/

# 攻撃面を減らす (3)

## sudo vs doas

- doas ― sudo の OpenBSD による再実装
- 複雑になるとバグ・脆弱性を招きやすい ― 機能を絞り、規模を減らすことで攻撃面を縮小する

| | sudo | doas |
|---|---|---|
| 設定ファイル | 179 行 | 14 行 |
| ソースコード | 178,381 行 | 782 行 |

```
# wc doas.conf
 14      82     577 doas.conf
# wc sudoers sudo.conf
 48     212    1499 sudoers
131     687    4384 sudo.conf
179     899    5883 total
# wc `find . -type f -name '*.[ch]' -print`
498    1580   11133 ./doas.c
 49     221    1393 ./doas.h
235     706    4943 ./env.c
782    2507   17469 total
# wc `find . -type f -name '*.[ch]' -print`
(466 個のソースファイル )
178381  650481 5720935 total
```

# 攻撃成功を困難に (1)

## W^X (write XOR execute)

- メモリへは「書き込み可」か「実効可」のどちらかのみ
- プログラムを「改変して実行」が困難に
- これを行うアプリもある (JIT 的な動作 ) ので、ファイルシステムのマウントオプションで制御

↓このようなメモリ取得は許可されない

```
void *mem = mmap(NULL, pagesize,
    PROT_READ | PROT_WRITE | PROT_EXEC,
    MAP_ANON | MAP_PRIVATE,
    -1, 0);
```

実行例 :

```
$ ./wxtest
mmap(RWX): Not supported
```

# 攻撃成功を困難に(2)

## Library order randomization

## Kernel relinking at boot

- ライブラリやカーネルの構成要素(オブジェクトファイル)を起動時に毎回ランダムな順番でリンクしなおす

  → 特定のアドレスを想定した攻撃が困難に

- Linux の ASLR, KASLR は「置き場所を変える」

  OpenBSD は「毎回中身をシャッフルして、置き場所も変える」

- 最近はカーネル、ライブラリ以外にも実行バイナリ(sshd, ssh-*, smtpd, httpd など)もランダムリンクするようになっている

# 攻撃成功時の影響を限定する (1)

## 権限放棄 / 権限分離

OpenSMTPD（メールサーバ）の例

- root: マスタープロセス
  - ポート 25 の bind や子プロセスの生成など、 root 権限が必要な操作
- _smtpq: キュー処理
- _smtpd: その他の処理
  - プロトコルの実装
  - 暗号化機能、など

```
USER     PID %CPU %MEM TT STAT    TIME COMMAND
root   81446  0.0  0.0 ?? Ip   0:00.01 /usr/sbin/smtpd
_smtpd 74529  0.0  0.0 ?? Ipc  0:00.01 smtpd: crypto (smtpd)
_smtpd 24094  0.0  0.0 ?? Ipc  0:00.01 smtpd: control (smtpd)
_smtpd 66206  0.0  0.0 ?? Ip   0:00.01 smtpd: lookup (smtpd)
_smtpd  7441  0.0  0.0 ?? Ipc  0:00.01 smtpd: dispatcher (smtpd)
_smtpq 44139  0.0  0.0 ?? Ipc  0:00.02 smtpd: queue (smtpd)
_smtpd 24311  0.0  0.0 ?? Ipc  0:00.01 smtpd: scheduler (smtpd)
```

# 攻撃成功時の影響を限定する (2)

## pledge / unveil システムコール

- pledge - 「利用する機能」をあらかじめ宣言 ( 誓約 )

  誓約していない機能を使うとエラー終了
- unveil - 「アクセスしてよいディレクトリ」をあらかじめ宣言。それ以外はアクセス不可

  例 :Firefox の Downloads

```
/* pledge の発効 (ping コマンド ) */
if (pledge("stdio inet dns", NULL) == -1)
    err(1, "pledge");
/* pledge していないファイル操作を実行 */
if ((logfile = fopen("/var/log/messages", "r"))
     != NULL) {
   printf("/var/log/messages opened\n");
   fclose(logfile);
}
```

# まとめ：OpenBSD のセキュリティ

- いままで紹介したセキュリティ技術

  → 先駆的であるが、今では多くの他の OS でも同様な実装あり
- OpenBSD に特徴的なことは？
  - **Secure by Default:** デフォルトで機能を有効に
  - 決定論的： ユーザが off できるスイッチを作らないことが多い
  - System Wide: 適用原則はシステム全体に - 例外を作らない
  - 妥協しない：「過去の経緯」や「大人の事情」に引きずられない

# 河豚板とは

- OpenBSD をベースにしたライブシステム (LiveDVD/LiveUSB)
  「このプロジェクトの第一目標は、
  **OpenBSDという優れた OS を皆さんに手軽に触れてもらう**ことです」

- 特徴
  - OpenBSD に極力手を加えず、そのまま提供
  - ライブシステムの実装に必要な機能にフォーカス
    - 環境の保存と復帰
    - システムの複製 ( リマスタリング )
    - デスクトップやネットワークなどの設定ウィザード
    - etc...

# 河豚板のセキュリティ（設計・運用）

- OpenBSD の Errata 対応
  - Errata ページの自動監視 → メール通知
  - 対応版作成 → 動作検証 → ミラーリング → リリース・広報
    3 プラットフォーム ×{ISO,IMG}×{UP,MP}CPU = 12 パス
    通常は 6H 以内程度で対応

- 河豚板 -current の提供 （amd64 用、 LiveUSB のみ ）
  - ユーザによる新機能検証用
  - 概ね、週一回リリース

# 河豚板のセキュリティ（技術面）

- OpenBSD に極力手を加えず、そのまま提供
  - OpenBSD のセキュリティ機能をほぼ**すべて継承**

- OpenBSD のセキュリティ機能に**上乗せ** ― live media の特性を利用
  - システムとユーザ環境の分離構造
    - システムは Read Only( 改変不可 )
    - ユーザ環境は MFS( メモリ上、揮発 )
    - 起動モードによるファイルシステム構成の調整
    - MFS の揮発性を利用した運用
  - live media の errata 適用

# 河豚板のセキュリティ(1)

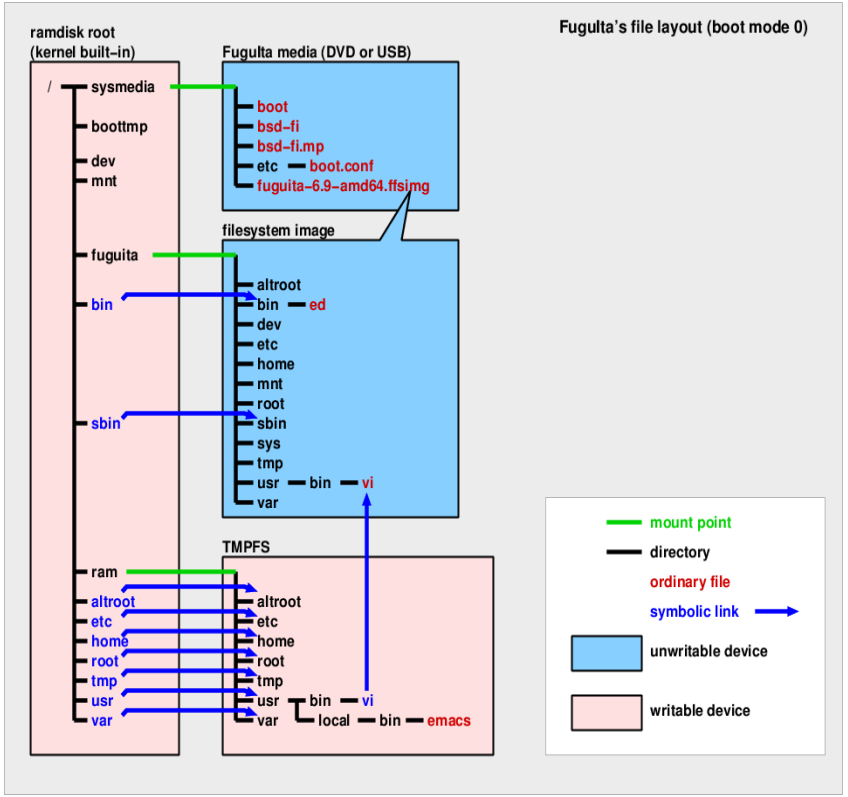
OpenBSD のセキュリティ機能は「ほぼ」すべて継承

- KARL (Kernel Address Randomized Link) だけは未実装
  - システム部分が Read Only( 書き込み不可 ) なため、
    リンクしたカーネルを次回起動用に保存できない
- 次善の策
  - リリース時にランダムリンク
    →カーネルに関係のない Errata 等でも、必ずランダムリンク
  - ビルドツールの公開 → 「どうしても KARL やりたい」人向け

# 河豚板のセキュリティ (2)

河豚板のファイルシステム構成

- /sysmedia
  - OS 本体 （ほぼ素の OpenBSD）
  - Read Only マウント
- /ram
  - 可変部分 （≒ ユーザ環境 ）
  - メモリ上、揮発性、 RW
  - usbfadm により永続化も可能




シンボリックリンクによる合成

# 河豚板のセキュリティ (3)

起動モードによる保護レベルの調整

- 起動モード 1

  /bin,/sbin,/usr は RO 、それ以外は RW … 実行バイナリ書き換え不可。 pkg_add も不可

- 起動モード 2

  全てが MFS 上、 RW 。ただし usbfadm sync しない限り揮発

- 起動モード 0 ( デフォルト )

  モード 1 と 2 の中間。 /usr 以下は symlink による shadow copy

# 河豚板のセキュリティ(4)

fiupdate (FuguIta Update) によるシステム更新

- リリースされた ISO イメージから、 OS のファイルシステムイメージ (fuguita-7.9-amd64.ffsimg など ) 、とカーネル (UP,MP) を取り出し、実行中のシステムに上書き→再起動
  - Errata 対象部分だけでなく、システム全体 ( カーネル含む ) をアトミックに更新
  - システムとユーザ環境は分離されているため、 /etc 以下のシステム設定や /home 以下のユーザデータなどは影響を受けない

# まとめと展望

OpenBSD

- セキュリティ機能の先駆的な実装
- 継続的・網羅的なアプローチ
- 妥協しない姿勢

河豚板

- OpenBSD セキュリティを可能な限り継承
- ライブシステム特有の機能を上乗せ
- メディアの RW/RO の利用
- システムとユーザ環境の分離

OpenBSD

- 今年の 3 月頃から Errata の頻度が増加
- 公式なアナウンスはないが、 AI の活用はやってるっぽい (OpenBSD Journal 等 )

河豚板

- Errata 対応へのリリース作業の省力化・自動化
- 作業時間の短縮

# ToDo

- ターゲット層の適切な設定
- 技術説明部分の粒度がバラバラ
- ビジュアルが少ない
- 発表時間による内容量の調整

- サイドストーリー（よもやま話）の追加
  - `strlcpy(OpenBSD, 1998)` vs `strncpy(Linux, 2026)` とか
  - 仮想化、コンテナ機能をセキュリティ的にどう見ているか、とか
-