# デバイスドライバの脆弱性調査と分析

藤澤　一樹 †　　大久保　隆夫 †　　田中　英彦 †

† 情報セキュリティ大学院大学
221-0835 神奈川県横浜市神奈川区鶴屋町 2-14-1

{mgs061101,dgs063104,tanaka}@iisec.ac.jp

**あらまし**　デバイスドライバは一般に特権で動作するので，セキュリティ上の致命的な問題となることが知られており，近年ではその脆弱性を狙った攻撃が増加している．そこで本稿では，デバイスドライバの脆弱性調査／分析から，デバイスドライバに起因するセキュリティ上の問題点とその原因を明らかにすることによって，既存研究ではこれを防げないことを述べる．さらに現状最も有効な対策として考えられているデバイスドライバの開発／テスト工程における脆弱性検査の限界を指摘し，運用時における保護の必要性を述べる．

# Survey and Analysis of Device Driver Vulnerability

Kazuki Fujisawa†　　Takao Okubo†　　Hidehiko Tanaka†

†Institute of Information Security
2-14-1 Turuya-cho, Kanagawa-ku Yokohama, Kanagawa, 221-0835 Japan

{mgs061101,dgs063104,tanaka}@iisec.ac.jp

**Abstract**　Generally, device drivers run in the privilege mode and it can be the critical problem for security. So crackers tend to aim the device driver's vulnerabilities in recent year. In this paper, we make survey device driver's vulnerabilities to point out the limitation of existing methods and argue about our dynamic approach that protect device drivers on running.

## 1　はじめに

デバイスドライバはデバイスを制御し，抽象化したインタフェースをアプリケーションに提供するためのソフトウェアである．近年では，デバイスが高機能化することによってデバイスドライバのコード量が増えたために，セキュリティ上のバグを持つ可能性が潜在的に高まっており，さらに Windows や Linux ではデバイスドライバが最高特権で動作するので，その問題はより深刻なものとなっている．加えて，Windows ではカーネルに対する仮想化機能やセキュリティ機能の追加など多様な用途向けにデバイスドライバを用いるので影響範囲も大きい．Linux では，最小特権のセキュリティ原則に基づいた強制アクセス制御機能を提供することで，攻撃者の root 権限奪取を防ぐ SELinux[1] があるが，その機能はユーザ空間向けのものであるため，デバイスドライバが動作するカーネル空間では効果を発揮できないという問題もある．

また，近年急速に普及している携帯電話や情報家電は Windows Mobile や Symbian OS，組み込み Linux などの汎用 OS を利用する傾向が強まっており，これらは個々のデバイス専用の OS に比べて多くのデバイスドライバを含んでいる．シンクライアントや VMM（Virtual Machine Monitor または Hypervisor）も同様にもデバイスドライバは必要である．

さらに，様々なセキュリティベンダーからサードパーティ製のデバイスドライバを狙った攻撃が今後，増加することが指摘されるようになり，デバイスドライバに関するセキュリティ上の問題が増えていくと考えられる．しかしながら，デバイスドライバに関する研究はこれら問題への対策よりも可用性を高める研究が主流である．

そこで本稿では，デバイスドライバの脆弱性調査／分析から，デバイスドライバに起因するセキュリティ上の問題点とその原因を明らかにすることによって，既存研究ではこれを防げないことを述べる．さらに現状最も有効な対策として考えられているデバイスドライバの開発／テスト工程における脆弱性検査の限界を指摘し，運用時における保護の必要性を述べる．

## 2 関連研究

本章では，デバイスドライバのバグを極小化する取り組み，特権で動作するデバイスドライバを極小化する取り組み，特権から完全にデバイスドライバを排除する取り組み，デバイスドライバを隔離する取り組みについて述べる．

### 2.1 バグの極小化

デバイスドライバそのものに存在するバグを出来る限り排除することでシステムへの影響を小さくする手法である．例として，Microsoft が提供しているデバイスドライバの検査ツールとデバイスドライバ署名 [2] を挙げる．デバイスドライバの検査ツールは，開発者に対して提供している Windows DriverKit（WDK）[4] というデバイスドライバ開発キットのテストツールである Static Driver Verifier（SDV）[5] のことで，主に関数のテストやアクセス制御の方法と関連する同期化などの項目を検査できる．もう一方のデバイスドライバ署名は，Microsoft がデバイスドライバを検証し，問題がなければ署名を行うものである．

また，Windows や Linux 等 OSS（Open Source Software）においても使えるソースコードの脆弱性検査等が行える Source Code Auditing Tool（以降 SCAT）が存在する．

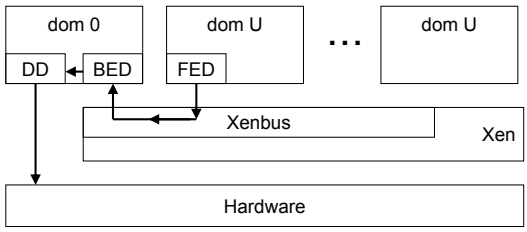


図 1: Xen アーキテクチャ

### 2.2 特権動作のデバイスドライバの極小化

Windows や Linux は特権でデバイスドライバを動作させていることについて既に述べたが，通常特権モードで動作するデバイスドライバを特権モード動作と非特権モード動作に分け，特権モード動作するデバイスドライバを少なくすることでデバイスドライバが原因で起きる問題の影響を小さくする手法である．この手法は Microsoft の Windows Vista でカーネルモードドライバとユーザモードドライバ [3] として実装されている．

### 2.3 特権動作からの排除

一般的に MINIX 3 をはじめとする Microkernel や Exokernel[6] においてはユーザ空間でデバイスドライバを動作させることができるため，特権からデバイスドライバを排除が可能である．

また，VMM 自体にデバイスドライバを持たせないことでデバイスドライバの脆弱性による影響を排除する手法がある．この手法はケンブリッジ大学で開発された VMM である Xen に実装されている．この Xen[7] は，既に研究されていた Microkernel，Exokernel らを踏まえており図 1 に示すようなアーキテクチャをとっている．Xen 自体はデバイスドライバを持たなく，管理ドメイン（dom 0）のデバイスドライバを利用する．dom 0 は Xen を管理する特権ドメインであり，非特権ドメイン（dom U）とは区別される．dom U からデバイスへのアクセス要求があった場合，必ず Frontend Driver（FED）が Xenbus を通って Backend Driver（BED）にアクセスし，dom 0 のデバイスドライバがデバイスへアクセスする構造をとっている．

### 2.4 隔離

Microkernel アプローチであれば，それぞれのサービスごとに隔離することが可能であるため，デバイスドライバが故障しても OS がクラッシュすることはない．しかし，モノリシックカーネルであるとデバイスドライバの故障が OS をクラッシュさせる．そこで，カーネル拡張機能を保護する仕組みとして研究されていた Nooks[8] をデバイスドライバに最適化した研究 [9] によってモノリシックカーネルであっても隔離と仮想的なデバイスドライバを用いることでデバイスドライバの故障から OS を保護できる．

## 3 調査と分析

この章では，デバイスドライバの脆弱性に起因する問題と，脆弱性の原因を明らかにするために，調査，分析を行った．デバイスドライバにおける脆弱性の調査概要及び，調査結果，分析結果について述べる．

### 3.1 調査概要

分析するにあたっては 2007 年 8 月 13 日の SecurityFoucus[10] の脆弱性情報を利用した．有名な脆弱性しか存在しないがデバイスドライバの脆弱性に起因する問題を整理し，脆弱性の原因を明らかにするためには十分であると考えた．デバイスドライバに関する脆弱性は 48 件該当したが，複数の脆弱性を一つの項目に書いているものは，複数の脆弱性を，それぞれ一つとしてカウントし，デバイスドライバというよりも，ライブラリや OS そのものの問題である場合は除外した．また，今回の調査はデバイスドライバの脆弱性に起因する問題と，脆弱性の原因を明らかにするために行うため OS やアプリケーション，デバイスの種類に関しては言及しない．

### 3.2 調査結果

SecurityFocus のサイトからデバイスドライバ以外のものを除外した結果，該当項目は 45 件になった．

調査結果における最も大きな分類は，設計ミス，実装ミス，実装項目の欠如，詳細不明の 4 通りにした．設計ミスについては，設計段階からセキュリティを考慮して作られておらず，問題を起こしたもの．実装ミスは，セキュリティに関する項目以外は全て実装済みのもの．実装項目の欠如についてはある部分の処理に対して何も書かれていないものと定義する．

次に最も大きな分類を小さく分類していくにあたってデータの取り扱い，権限の取り扱い，入力値の取り扱い，バッファの取り扱い等にした．また，矢印（→）は，分類を受けて何が原因となったかを指名している．表 1 に調査結果を示す．

また，この調査では Windows Vista におけるデバイスドライバの脆弱性は存在しなかった．その理由として，発売時からの時間が短いこと，SCAT，SDV，署名，ユーザモードデバイスドライバとマイクロカーネルアプローチがとられていることが考えられるが，Windows XP や Windows 2000 におけるデバイスドライバの脆弱性はセキュリティ対策ソフトウェアが用いるデバイスドライバの場合が多く，そうなれば Windows Vista でも起きるため時間の問題だと考えられる．

### 3.3 分析結果

根本的な原因として，もっとも高い割合を示すのは図 2 に示すように圧倒的に実装ミスが多い．その中でも，図 3 から,「入力値の取り扱い」,「バッファ不足」,「権限の取り扱い」で 70%を占める．さらに「メモリの取り扱い」,「予期しない動作」を含めると 88%という割合が多くのプログラミングで共通する実装ミスにあたり，上位 3 件である「入力値の取り扱い」,「バッファ不足」,「権限の取り扱い」という脆弱性によって起きる問題は，権限昇格／任意のコードが実行可能／クラッシュ／ DoS という非常にクリティカルな問題が引き起こされることがわかる．

一般にアプリケーションでは，Buffer Overflow の脆弱性が多数を占めるが，デバイスドライバでは，入力値の取り扱いや権限の取り扱いミスが存在する．

表 1: デバイスドライバにおける脆弱性のまとめ

| 分類 | 原因 | 結果 |
|---|---|---|
| 設計ミス | データの取り扱い → Information Disclosure | データリーク |
| | データの取り扱い | セキュリティソフトの回避可能 |
| | 特定の条件 | システムクラッシュ |
| 実装項目の欠如 | データの取り扱い → 最小データの処理 | データリーク |
| | その他→ Flag の未実装 | 不明 |
| 実装ミス | 入力値の取り扱い | 権限昇格／任意のコードが実行可能／クラッシュ／ DoS ／データリーク |
| | → Integer Overflow | 権限昇格／任意のコードが実行可能／クラッシュ |
| | バッファ不足 → Buffer Overflow | 権限昇格／任意のコードが実行可能／クラッシュ |
| | 権限の取り扱い | 権限昇格／任意のコード実行可能／ DoS |
| | メモリの取り扱い | クラッシュ／権限昇格 |
| | → メモリリーク | DoS |
| | → Memory Disclosure | データリーク |
| | 予期しない動作 | クラッシュ／ DoS |
| | Secure Code Audit | 権限昇格／ Buffer Overflow ／ DoS ／クラッシュ |
| | データの取り扱い → データの保存方法 | データリーク |
| | バグ | DoS |
| 詳細不明 | ——— | DoS ／権限昇格 |
| | ——— → DRM の実装 | X のクラッシュ |
| | ——— → Information Disclosure | クラッシュ |
| | データの取り扱い → 特定の条件 | クラッシュ |

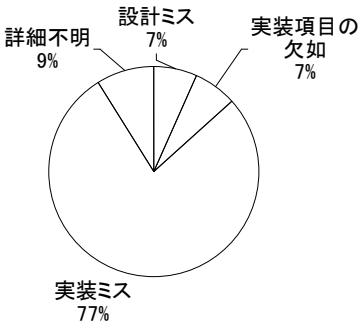


図 2: 根本的な原因の割合

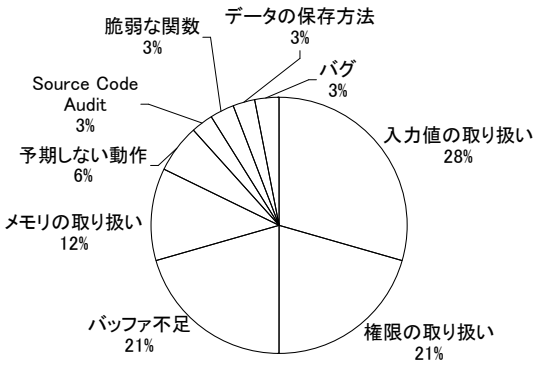


図 3: 実装ミスにおける原因の割合

## 4 不完全な保護方法

2 章において，現在，行われている取り組みを述べた．3 章において，デバイスドライバの脆弱性に起因する問題と脆弱性の原因を明らかにし，デバイスドライバの脆弱性が如何に脅威であるかを述べた．この章では，現在，行われている方法では根本的な解決にはならないことを示す．

脆弱性検査ツールは，有効であると考えられるが SDV によって検査できるのは Windows ドライバモデル (WDM) 使用ルール違反が無いかどうかを検査しているからであり，WDM のような，デバイスドライバに対しての決まりが存在していない場合は不可能であり，WDM 自体にセキュリティ上の問題があった場合などは修正も困難にする．また，デバイスドライバから完全に脆弱性を排除することは非常に困難である．デバイスドライバ署名も Windows Driver Foundation(WDF) が存在していることと，OS が Microsoft のみで開発されているからであり，OSS で署名を行うとすると誰が署名するのかが難しく困難である．また，SCAT である OSS の

RATS - Rough Auditing Tool for Security[11] を利用し，実装ミスの上位 3 件から数件をチェックしたが,「バッファ不足」に起きる既知の Buffer Overflow は検知できたものと未検知のもの,「権限の取り扱い」については検知不可,「入力値の取り扱い」の Integer Overflow については検知というように，SCAT は，ウィルス対策ソフトと同様にシグネチャ方式であるため既知のものしか検知できないだけでなく，既知，未知を含めて全て検出できるわけではない．また，次のようにプログラムのコンテキストに依存するため，前提が必要なものや，トリガーとなる関数がユーザ定義のものなどは検出するのが難しい．特にデバイスドライバは，そのデバイス固有のコードが存在するため一般的な監査では完全に脆弱性を取り除くことは不可能である．

既に実装されているカーネルモードドライバとユーザモードドライバは，特権で動作しているデバイスドライバがあるため根本的な解決にならない．

Microkernel を採用している Windows 2000 や XP，Vista であっても特権でデバイスドライバを動作させているため根本的な解決にはならず，Exokernel アーキテクチャを持つ OS は一般的に使われていない．また，Xen のように VMM にデバイスドライバを持たせない方法は，VMM を安全にする方式である．しかしながら，Xen においては Xen を管理できる dom 0 のデバイスドライバを用いているため，デバイスドライバの脆弱性を突かれて，dom 0 をリモートコントロールできるようになると，Xen のリモートコントロールが可能となる．Nooks や，その後の研究のようにデバイスドライバを隔離し保護する仕組みは非常に有効であると考えられるが，これらの研究はデバイスドライバが故障した場合に OS をクラッシュさせないために開発されたものであり脆弱性の保護には使えない．

## 5　考察

3 章と 4 章においてデバイスドライバの脆弱性に起因する問題と脆弱性の原因を明らかにし，現状の保護方法では防げないことを述べた．そこで本章では現状における対策を述べ，その問題点について考察する．

ウォータフォールモデルのソフトウェア開発であれば設計→実装→テスト→運用という流れになる．近年では，それぞれの工程において脆弱性検査が必要だと言われるが，全てに対して脆弱性検査をすると非常にコストがかかるため運用時に IDS や Firewall，sandbox などで後付的にセキュリティ機能を付加する場合が多い．

デバイスドライバに関しては特に運用時に保護する仕組みはなく，対策として実装時とテスト時における脆弱性検査が重要になる．この実装時とテスト時に防ぐことが可能な脆弱性は「実装項目の欠如」と「実装ミス」であり，全体の 84%以上を占める．そのため実装時は，原因として述べた「入力値の取り扱い」,「バッファ不足」,「Source Code Audit」,「脆弱な関数」で 55%の問題を防ぐことができる．補助的に SCAT を用いることで，危険な関数の仕様や既知の脆弱性については，ある程度検知することができるため人間が目だけで検査するよりも脆弱性を無くす効率を挙げることができる．次の段階であるテストでは,「予期せぬ動作」を行わないか,「メモリの取り扱い」によって別の空間への上書きやメモリリークを起こさないか,「権限の取り扱い」,「データの保存方」に問題はないかという点でテストすることで 42%を防ぐことができる．このように開発工程とテスト工程における脆弱性検査において理想的にいけば実装ミスの 97%，全体で 81%が防ぐことができる．

しかしながら，既に述べた通り，SCAT は全てのソースコードにおける問題を全て検知できず，あくまでも最後は人が見て判断する必要がある．場合によっては脆弱でないケースを脆弱であると通知することもあるため知識のない者が使用して脆弱性を発見することは困難である．また，テスト時においても全てのテスト項目を網羅することは不可能である．そのため，現状の対策として行える開発，テストの脆弱性検査だけでは限界があるという問題がある．

そこで運用時に保護する仕組みが必要となる．この運用時に保護する仕組みはデバイスドライバが動作中に脆弱性を突かれた場合でも問題を

起こさないようにする方法である．例えば，各デバイスドライバを sandbox の中で動作させるなどが考えられるし，デバイスドライバに Secure OS と同様のアプローチを取り入れることによって，脆弱性を持っていても問題を起こさず動作させる研究 [12] もある．このように，デバイスドライバであっても運用時に保護する仕組みがなければデバイスドライバの脆弱性からシステムを保護することはできない．

## 6 まとめと今後の課題

本稿では，デバイスドライバの脆弱性の危険性を指摘し，調査，分析を経て脆弱性に起因するセキュリティ上の問題と，脆弱性の原因を明らかにした．そして，現在の保護方法では防げないこと指摘し，対策として実装時，テスト時の脆弱性検査が重要であることを述べた．さらに，問題点として，それらの限界を指摘し，運用時に保護する仕組みの必要性を述べた．

今後の課題としては，運用時にデバイスドライバを動的に保護できることを示していきたい．

## 参考文献


1) SELinux
Stephen D. Smalley, “NSA SecurityEnhanced Linux”, Ottawa Linux Symposium BOF 2006.

2) Microsoft, ロゴ及び WHQL テスト
http://www.microsoft.com/japan/whdc/GetStart/default.mspx

3) Windows Driver Foundation（WDF）
http://www.microsoft.com/japan/whdc/driver/wdf/default.mspx

4) Windows Driver Kit
http://www.microsoft.com/japan/whdc/devtools/wdk/default.mspx

5) Static Driver Verifier
http://www.microsoft.com/japan/whdc/devtools/tools/sdv.mspx

6) MIT Exokernel Operating System
http://pdos.csail.mit.edu/exo.html

7) Keir Fraser, Steven Hand, Rolf Neugebauer, Ian Pratt, Andrew Warfield, Mark Williamson, “Safe Hardware Access with the Xen Virtual Machine Monitor”, OASIS ASPLOS 2004 workshop.

8) Michael M. Swift, Brian N. Bershad, and Henry M. Levy, “Improving the Reliability of Commodity Operating Systems”, ACM Transactions on Computer Systems, 22(4), Nov. 2004.

9) Michael M. Swift, Muthukaruppan Annamalai, Brian N. Bershad, Henry M. Levy, “Recovering Device Drivers”, Proceedings of the 6th ACM/USENIX Symposium on Operating Systems Design and Implementation, San Francisco, CA, Dec. 2004.

10) SecurityFocus
http://www.securityfocus.com/

11) RATS - Rough Auditing Tool for Security
http://www.fortifysoftware.com/security-resources/rats.jsp

12) 藤澤 一樹, 橋本 正樹, 宮本 久仁男, 金 美羅, 辻 秀典, 田中 英彦, “仮想マシンモニタにおけるデバイスドライバ安全性向上に関する提案”, FIT2007 第 6 回情報科学技術フォーラム Sep. 2007.