IPA Technical Watch

IPA テクニカルウォッチ
製品の品質を確保する
「セキュリティテスト」に関するレポート

～ 修正費用の低減につながるセキュリティテスト「ファジング」の活用方法とテスト期間に関する考察 ～

独立行政法人情報処理推進機構
セキュリティセンター

IPA テクニカルウォッチ：

製品の品質を確保する「セキュリティテスト」に関するレポート

～ 修正費用の低減につながるセキュリティテスト「ファジング」の活用方法とテスト期間に関する考察 ～

目次

IPA テクニカルウォッチ：

製品の品質を確保する「セキュリティテスト」に関するレポート

～ 修正費用の低減につながるセキュリティテスト「ファジング」の活用方法とテスト期間に関する考察 ～

2012 年 9 月 20 日

IPA（独立行政法人 情報処理推進機構）
セキュリティセンター

本書の要旨

製品の品質を確保するセキュリティテストには、「ファジング（英名：fuzzing）」というテスト手法がある。この「ファジング」は Microsoft 社などの大手ソフトウェア企業が採用しており、品質の確保に十分な成果を挙げている。また制御システムの分野においては、評価・認証の標準化において「ファジング」が要件の一つとなる流れである。

本書では、「ファジング」の概要・効果・活用事例、そして活用方法をまとめている。特に、「ファジング」導入を検討する際に必要となる情報（費用対効果の統計情報、IPA の活動における実測値など）に焦点を当てている。本書が製品開発ライフサイクルにおいて「ファジング」を活用する一助となれば幸いである。

本書の想定読者

製品開発企業の経営者・品質保証部門の部門長
（製品開発ライフサイクルの[テスト]工程に責任を持つ方）

本書の構成

【知る】(*)

ファジングそのものや効果、活用事例などをまとめています。

1章. セキュリティテストの重要性
2章.「ファジング」の概要
3章. ファジングによるバグや脆弱性のリスクの低減

【使う】

ファジングを活用するうえで検討する課題（費用とテスト期間）、課題をふまえたアプローチの提案をまとめています。

4章.「脆弱性検出の普及活動」に基づくファジングの活用方法

(*):
IPAの「ファジング活用の手引き」と内容が一部重複します。すでに「ファジング活用の手引き」を読んだ方は、【使う】から読みすすめても構いません。

本書の用語

用語	説明
製品	ブロードバンドルータやデジタルテレビなどの組込み機器、ソフトウェア製品など市場で販売されているものをまとめて「製品」と定義する。
バグ	本書では、製品が仕様通りに動作しなくなる等の問題を「バグ」と定義する。
脆弱（ぜいじゃく）性	本書では､製品を強制的に再起動させてしまう等のセキュリティ上の問題を｢脆弱性」と定義する。この定義は､「情報セキュリティ早期警戒パートナーシップ[1]」におけるガイドラインに沿っている。

[1] IPA：「情報セキュリティ早期警戒パートナーシップガイドライン」
http://www.ipa.go.jp/security/ciadr/partnership_guide.html

1　セキュリティテストの重要性

製品開発においては、製品に要求される機能を実現するとともに、「その製品がどの程度の処理能力をもつか」など明確に定義することが難しい要求（非機能要求）を実現する必要がある。機能要求とともに非機能要求を十分に実現することで、高い品質を達成できる。非機能要求にも様々な項目があるが、その中の一つに「セキュリティ」という項目がある[2]。

近年、製品出荷後に製品におけるセキュリティ上の問題（脆弱（ぜいじゃく）性）が発覚して、それにより製品利用者が被害を受けてしまう事例が多くなっている。非機能要求に「セキュリティ」という項目が設けられていることから、「製品における脆弱性の解消」も品質の一部であると考える。

機能要求や他の非機能要求で求められる品質を確保するためには、該当するソフトウェアテストを実施する。非機能要求における「セキュリティ」でも、それは同様である。ソフトウェアテストの中には、特に脆弱性の発見に焦点を当てた「セキュリティテスト」がある（図 1）。このセキュリティテストには「ソースコードセキュリティ検査」や「ファジング」がある。

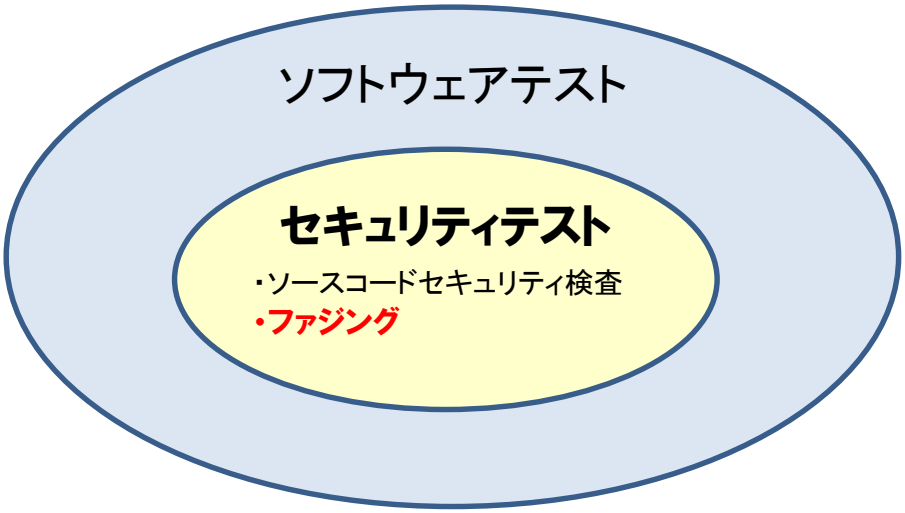


図 1 ソフトウェアテストにおける「セキュリティテスト」

本書ではファジングの普及啓発を目的とした「脆弱性検出の普及活動[3]」の実績をもとに、製品の品質を確保するファジングの活用方法と効果について報告している。なお、「ソースコードセキュリティ検査」については、2011 年 11 月に公開した「IPA テクニカルウォッチ：『ソースコードセキュリティ検査』に関するレポート[4]」を参照していただきたい。

---

[2] IPA が公開している「非機能要求グレード」でも、スコープの一つに「セキュリティ」を挙げている。http://sec.ipa.go.jp/reports/20100416.html

[3] http://www.ipa.go.jp/about/press/20110728.html

[4] http://www.ipa.go.jp/about/technicalwatch/20111117.html

2　「ファジング」の概要

2.1 ファジングとは

ファジングとは、製品などに何万種類もの問題を起こしそうなデータ(例:極端に長い文字列)を送り込み、製品の動作状態(例:製品が異常終了する)からバグや脆弱性を発見するテストである。

ファジングのイメージ図が図 2 となる。図 2 では[問題を起こしそうなデータ]を 1 つ送るたびに対象製品が応答することを確認している。[問題を起こしそうなデータ](1)、(2)と順番に送り、[問題を起こしそうなデータ](12345)を送ったところで対象製品から応答がなくなってしまった。対象製品のこの挙動を異常とみなすと、[問題を起こしそうなデータ](12345)を送ったところでバグまたは脆弱性の発見となる。

ファジングとは、対象製品に問題を起こしそうなデータを片っ端から送ってみて、力ずくでバグや脆弱性を発見するテストと言える。

図 2 ファジングのイメージ図

2.2 ファジングで得られる効果

製品開発において、ファジングを活用することで図 3 の 2 つの効果が得られる。①についてはテスト全般に言えることだが、②については特にファジングで得られる効果と言える。

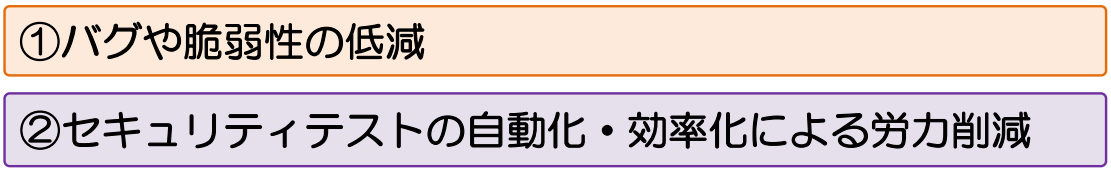


図 3 ファジングで得られる 2 つの効果

【①バグや脆弱性の低減】

ファジングは、ファジングを実施するための専用ツールである「ファジングツール」を使って実施するため、何度も繰り返し実施することが比較的容易である。繰り返し実施し問題点の修正を積み重ねることで、バグや脆弱性の低減が期待できる。また、ファジングのテストデータをファジングツールが機械的に作成するため、製品の設計者や開発者が想像しえないようなデータによって「想定外」の問題を発見できる可能性がある。

【②セキュリティテストの自動化・効率化による労力削減】

多くのファジングツールは人の操作をほとんど必要としないため、比較的楽な工程でバグや脆弱性を発見できると言える。また、同じような入力データを想定した製品に、テストデータを流用したり、再利用することが容易であるため、同種の製品をテストする場合にはセキュリティテストの労力や工数の削減にもつながる。

2.3 開発ライフサイクルにおける活用

では、製品の開発ライフサイクルにおいて、どの工程でファジングが活用されているのか。ファジングを採用している大手ソフトウェア製品開発企業の取り組みを基に説明したい。

なお、本書で「開発ライフサイクル」といった場合、「要件定義」から「設計」、「実装」、「テスト」、「運用/利用」、「廃棄(サービス終了)」に至る一連の工程を指す(図 4)。

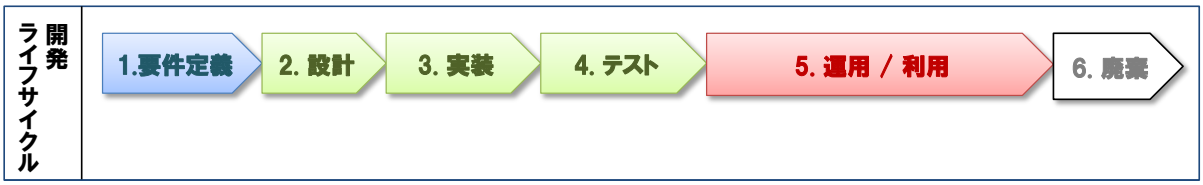


図 4 製品の開発ライフサイクル

2.3.1 海外の製品開発企業でのファジング活用

海外の製品開発企業のうち、ソフトウェア製品の開発ライフサイクルでファジングを活用している企業には Microsoft 社、Cisco Systems 社、Adobe Systems 社などがある。これら 3 社は、ファジングを活用することで、表 1 のような成果を挙げている[5]。

表 1 製品開発企業におけるファジング活用の成果

企業名	成果
Microsoft 社	「Office 2010」に対してファジングを実施して、1800 件の問題点を発見した。
Cisco Systems 社	同社製品に対してファジングを実施して、製品の問題を発見している。
Adobe Systems 社	「Adobe Flash Player」に対してファジングを実施して、80 件の問題点を発見した。

Microsoft 社、Cisco Systems 社、Adobe Systems 社は各社の開発ライフサイクルにファジングを導入して実施している。各社の開発ライフサイクルによると、これら 3 社は開発ライフサイクルの[テスト]工程でファジングを活用している(図 5 から図 7 の赤枠部分)。

http://www.microsoft.com/downloads/ja-jp/details.aspx?familyid=918179a7-61c9-487a-a2e2-8da73fb9eade から引用

図 5 Microsoft 社の開発ライフサイクルでのファジングの活用

[5] インターネットに公表されている情報を基にまとめた。

http://www.cisco.com/web/about/security/cspo/csdl/process.html#~tab-7 から引用

図 6 Cisco Systems 社の開発ライフサイクルでのファジングの活用

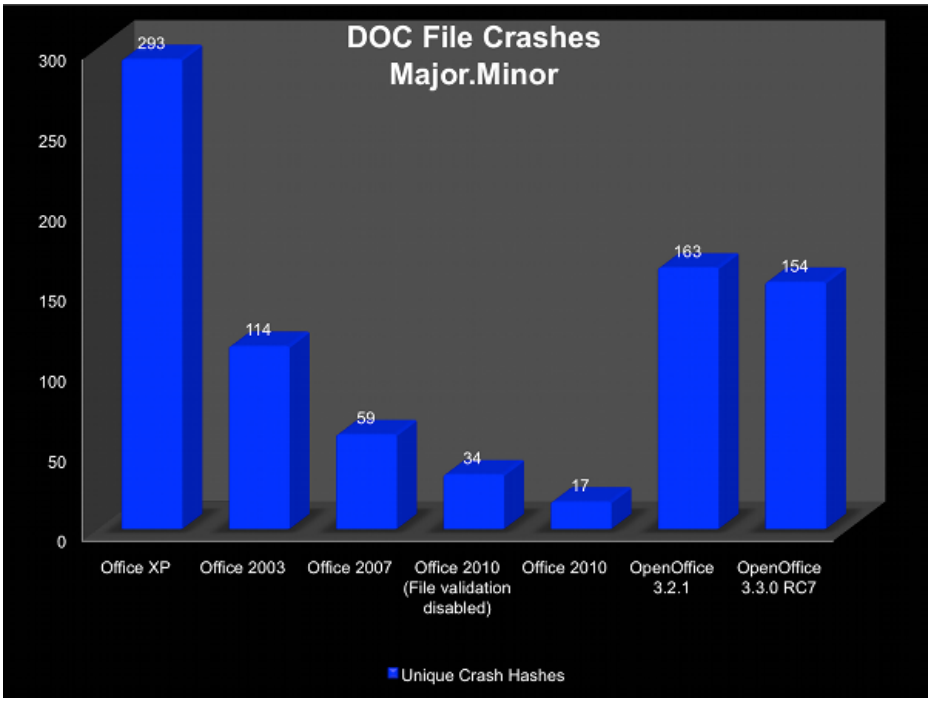
http://www.adobe.com/security/pdfs/privacysecurity_ds.pdf から引用

図 7 Adobe Systems 社の開発ライフサイクルでのファジングの活用

特に Microsoft 社においては、2002 年頃から大小あらゆる製品の開発ライフサイクルにおいて、ファジングを実施している[6]。この効果は、第三者機関である米国 CERT/CC（CERT Coordination Center）の実証結果[7]にも表れている。

図 8 は、2010 年 11 月に CERT/CC が Microsoft 社「Microsoft Office」、Oracle 社「OpenOffice」に対して細工したファイルを読み込ませるファジングを実施して、その結果発見した問題（バグや脆弱性）数を示している。Microsoft 社がファジングを実施する 2002 年以前に開発された「Office XP」では 293 件の問題を発見したが、2002 年以降に開発された「Office 2003」、「Office 2007」、「Office 2010」では同テストで発見した問題数が明らかに減少していることが分かる。

Microsoft 社が長年セキュリティを意識した開発ライフサイクルに取り組んだ結果だと考えるが、ファジングも製品の品質を確保する一助となっていると言えるだろう。



http://www.cert.org/blogs/certcc/2011/04/office_shootout_microsoft_offi.html から引用

図 8 CERT/CC によるファジング結果（2010 年 11 月）

[6] gihyo.jp：ソフトウェアの脆弱性検出におけるファジングの活用第 4 回　企業におけるファジングの導入事例
http://gihyo.jp/dev/feature/01/fuzzing/0004

[7] CERT/CC Blog: A Security Comparison: Microsoft Office vs. Oracle Openoffice:
http://www.cert.org/blogs/certcc/2011/04/office_shootout_microsoft_offi.html

2.3.2　日本の製品開発企業でのファジング活用

IPA がファジングを活用している日本の製品開発企業にヒアリングしたところ、興味深い結果が得られた。ファジングを活用している日本の製品開発企業 5 社に IPA がヒアリングしたところ、5 社のうち 4 社（うち 1 社は複数工程）は 2.3.1 節と同様に開発ライフサイクルの[テスト]工程でファジングを活用していることが分かった（図 9）。

図 9 で、5 社のうち 2 社が本来の活用工程である[テスト]工程ではなく、[実装]工程でファジングを活用していた点に注目してほしい。具体的には、別の企業に委託開発したプログラムを受領するときに「ファジングで問題がでないこと」を受け入れ条件としていた。

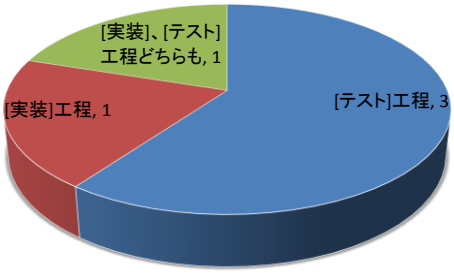


図 9 日本企業におけるファジング活用工程（IPA ヒアリング結果に基づく）

一般的に言われることだが、開発ライフサイクルの後の工程になるに従い、検出したバグや脆弱性を修正する費用は増大する（図 10 はそのイメージ図）。[設計]工程でバグや脆弱性を発見して修正する費用を基準（1 倍(X)）とすると、[実装]工程（図 10 では[開発]工程）では 6.5 倍(X)、[テスト]工程では 15 倍(X)というイメージで増加する。

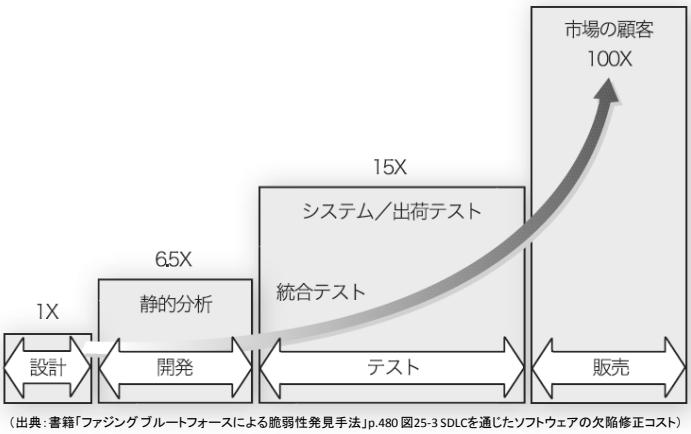


図 10 開発ライフサイクルにおける脆弱性等の修正費用（イメージ図）

製品開発企業では開発ライフサイクルの[テスト]工程でファジングを活用しているのが主流となっているが、[実装]工程での利用も 2 社が挙げている。IPA の活動の中でも検証したことであるが、[テスト工程]以外の工程でもファジングを活用できると考える。これについては、4.4.2 節で紹介する。

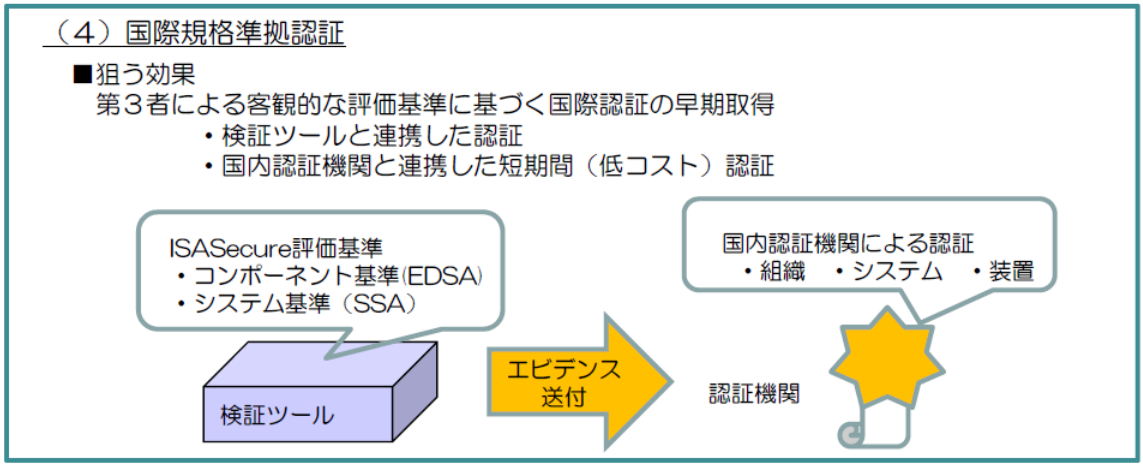
2.4 制御システム分野における評価・認証の標準化の流れ

様々あるコンピュータシステムの分野の中には、ファジングが評価・認証の標準化に組み込まれつつある分野がある。それが制御システム[8]の分野である。

制御システムの分野においては 2010 年に発覚したウイルス Stuxnet などの事件[9]もあり、セキュリティの意識やニーズが非常に高まっている。この制御システムのセキュリティ標準を規定し、それに沿った評価や認証などを確立する動き[10]が進められている（図 11）。分野共通の基準の策定が進められているものに、IEC62443 と ISASecure EDSA(Embedded Device Security Assurance)が挙げられる。これらでは、装置に内蔵されるシステムのセキュリティ機能、開発プロセス、テスト仕様などが規定されている。

評価・認証が先行している ISASecure EDSA では認証レベルによって、開発プロセスでの「Security Integration Test」でファジングが要件として定められている。またテスト仕様での「Robustness Test」において通信プロトコルにおけるファジングの要件が定められている。このテスト仕様は、ISASecure EDSA の認証を取得する場合には必須となっており、ファジングが標準化の流れの中に取り込まれている。以上のことから、一部の分野においては認証を取得するうえでファジングの実施が必須となってきている。

制御システム分野における評価・認証の標準化の流れについては、IPA の「2010 年度 制御システムの情報セキュリティ動向に関する調査報告書[11]」でもまとめている。



http://css-center.or.jp/pdf/cssc-activity.pdf から引用

図 11 制御システムセキュリティセンター（CSSC）事業内容の一つ「国際規格準拠認証」

8 生産ラインや輸送系システム、電力、ガス、水道などといった工業基盤や社会インフラに関わるシステム群を指す。

9 IPA：IPA テクニカルウォッチ：『新しいタイプの攻撃』に関するレポート
http://www.ipa.go.jp/about/technicalwatch/20101217.html

10 技術研究組合 制御システムセキュリティセンター（CSSC）
http://css-center.or.jp/

11 IPA：「2010 年度 制御システムの情報セキュリティ動向に関する調査」報告書の公開
http://www.ipa.go.jp/security/fy22/reports/ics_sec/index.html

3　ファジングによるバグや脆弱性のリスクの低減

3.1 バグや脆弱性におけるリスクと影響

まずバグや脆弱性が残ったまま製品を出荷した場合、どのようなリスクがあり、そのリスクが顕在化したときにコスト管理の観点からどのような影響があるかを考えてみよう。製品出荷後にバグや脆弱性が発覚した場合には表 2 のようなリスクと影響を想定できる。

表 2 ファジングを実施しない場合のリスクと影響

項目	説明
リスク	● バグに起因する製品の不安定な動作 ● 脆弱性を悪用した攻撃
コスト管理上の影響	● バグや脆弱性の改修から対応完了までの費用発生 ※製品によって、この費用が異なる。

表 2 のリスクが顕在化した場合、製品開発企業はそのバグや脆弱性を改修する必要がある。この改修に掛かる費用は製品種別によって変わってくるだろう。「ソフトウェア製品」、「組込み機器」、「制御システム」の 3 種を例にとり、バグや脆弱性の改修から対応完了までに掛かる費用を図 12 にまとめた。

単純にバグや脆弱性を改修する費用だけではなく、その製品の利害関係者への説明や被害が出た場合の回収・賠償などの費用も考慮する必要があるだろう。加えて、その製品の利用者数にともない、より多くの費用が掛かることが見込まれる。

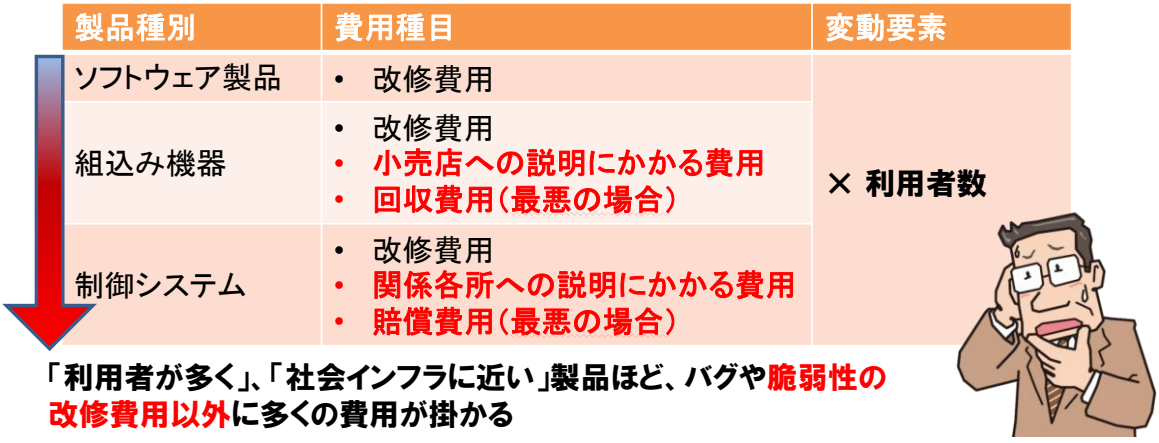


図 12 製品ごとの改修から対応完了までに掛かる費用種目

3.2 ファジングによるリスクの低減

3.1 節のリスクが顕在化する可能性を低減するために、テスト手法の一つとしてファジングを活用できる。ファジングを実施することで、これまでのテストで発見できなかったバグや脆弱性を発見できるようになる。

図 13 のイメージ図で例示しよう。製品開発ライフサイクルの[テスト]工程でテスト①、②, ③を実施したとする。バグや脆弱性のなかにはこれらのテストでは発見できないものがある（灰色のバグや脆弱性）。ファジングを実施することで、それらを製品出荷前に発見できるようになる。結果として、製品出荷後に該当製品に残っているバグや脆弱性が少なくなり、3.1 節のリスクが顕在化する可能性を低減できる。

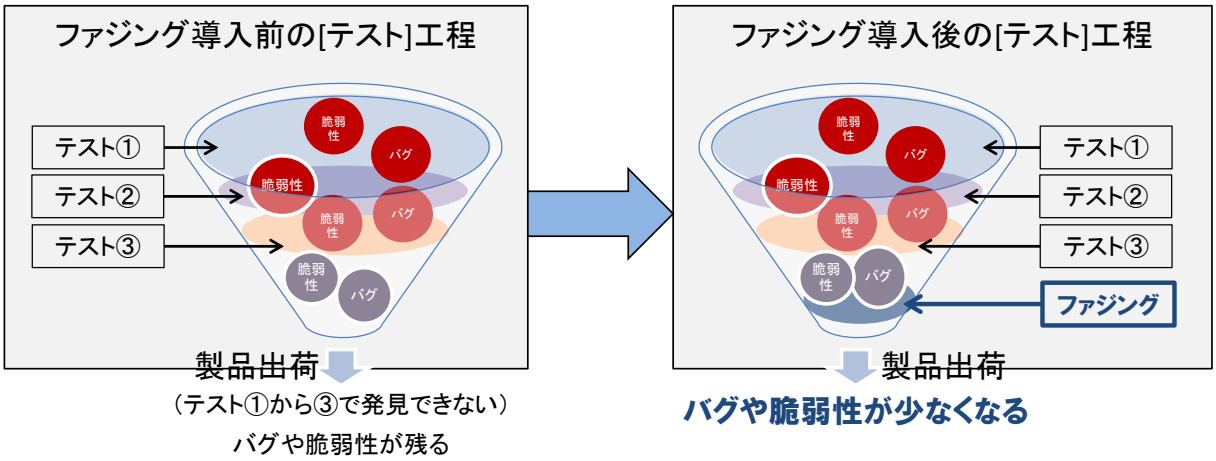


図 13 ファジング導入前後のイメージ図

ファジングには、「テスト対象の製品さえあれば誰でも実施できる」という特長がある。そのため、製品を開発した企業以外に「研究者」や「攻撃者」が積極的に活用して、バグや脆弱性を発見している（図 14）。製品開発企業に報告する「研究者」に脆弱性を発見されればよいが、「攻撃者」に脆弱性を発見された場合、その脆弱性が攻撃に悪用される可能性がある。

ファジングツールには商用製品もあるが、オープンソースソフトウェアとして開発されているツールも数多く存在する。このため、製品開発に携わっている方がファジングを活用すれば、同ツールで発見できるバグや脆弱性を製品出荷前に発見して修正できると言える。

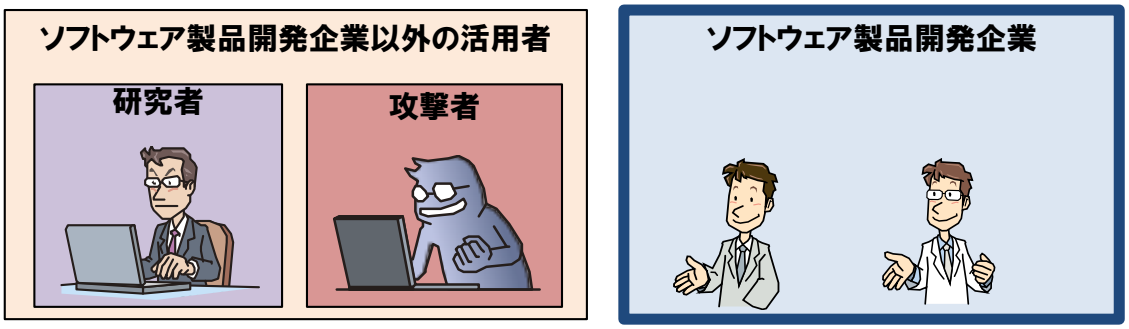


図 14 ファジングを活用する人たち

4 節では、IPA の「脆弱性検出の普及活動」の実績に基づく製品開発ライフサイクルでのファジングの活用方法を説明する。

4　IPA の活動で得られたファジングの活用方法に関する考察

4.1 「脆弱性検出の普及活動」の概要と活動実績

IPA では、2011 年 8 月からファジングの普及啓発を目的とした「脆弱性検出の普及活動」を開始した（活動のイメージ図は図 15）。この活動では、本書執筆時点では主にブロードバンドルーターやデジタルテレビなどの市販製品にファジングを実施して、ファジングの有効性実証などを実施している。2012 年 3 月に活動の知見をまとめた「ファジング活用の手引き[12]」および「ファジング実践資料」を公開した。

2012 年 6 月までの IPA の活動において、組込み機器 17 機種にファジングを実施したところ、合計 24 件の脆弱性を発見した。発見した脆弱性については適宜製品開発者に連絡している。

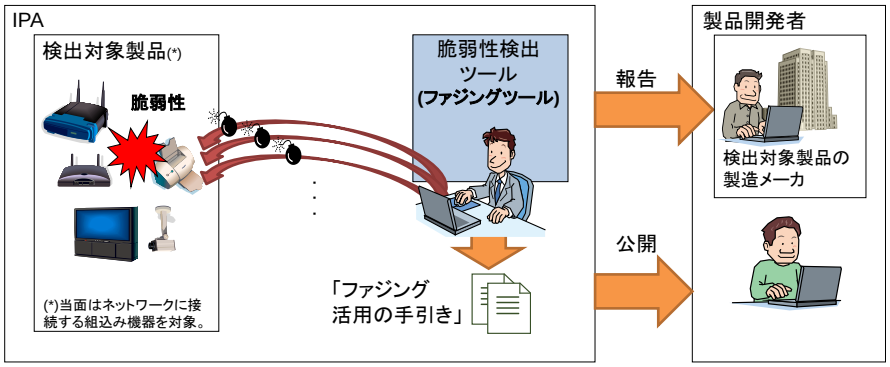


図 15 IPA の「脆弱性検出の普及活動」

4.2 開発ライフサイクルへのファジングの導入に関する考察

本節では、これまでの説明をふまえて製品の開発ライフサイクルにファジングを導入していない製品開発企業がファジングを活用する方法を考察してみたい。

製品開発ライフサイクルに企業が新たに「ファジング」を導入する場合には検討する課題が 2 つあると考える（図 16）。検討課題の 1 つ目は何よりも「ファジングの導入・運用に掛かる費用」であろう。検討課題の 2 つ目は「テスト期間への影響」だ。製品開発企業にとって、この 2 つの検討課題に納得できる解決策が与えられると、品質を確保するためにファジングが有効な選択肢となる。

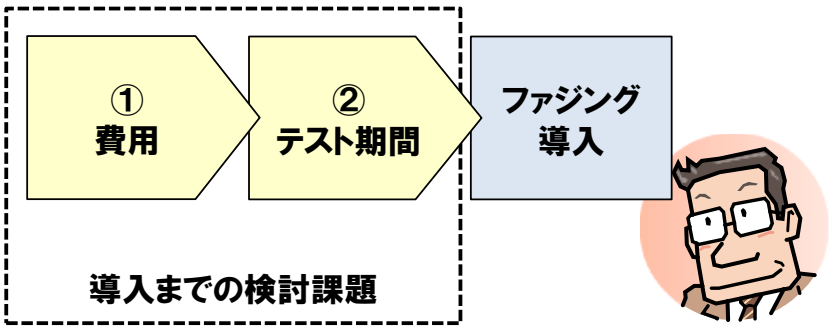


図 16 ファジング導入までの検討課題

[12] IPA：ファジング活用の手引き
http://www.ipa.go.jp/security/vuln/fuzzing.html

4.2.1　ファジング導入・運用に掛かる費用

3.2 節で述べたようにファジングを「リスクが顕在化する可能性を低減できるテストの一つ」として捉えると、製品出荷後に（ファジングで発見できる）問題が発覚してその対応に掛かる費用とファジングの導入・運用に掛かる費用との比較がポイントになる（図 17）。

製品出荷後に問題が発覚してその対応に掛かる費用は、製品や企業規模によって異なるため一概には言えない。だが、3.1 節で説明したように「利用者が多く」、「社会インフラに近い」製品になるほど、その費用は増大する。極端な例では約 120 億円を掛かってしまった事例[13]もある。一方で、そのリスクを低減するために、商用のファジングツールを購入して本格的にファジングを実施する場合、導入費用と運用費用（ツールの保守費用）をあわせて数百万から数千万円の費用が掛かる。

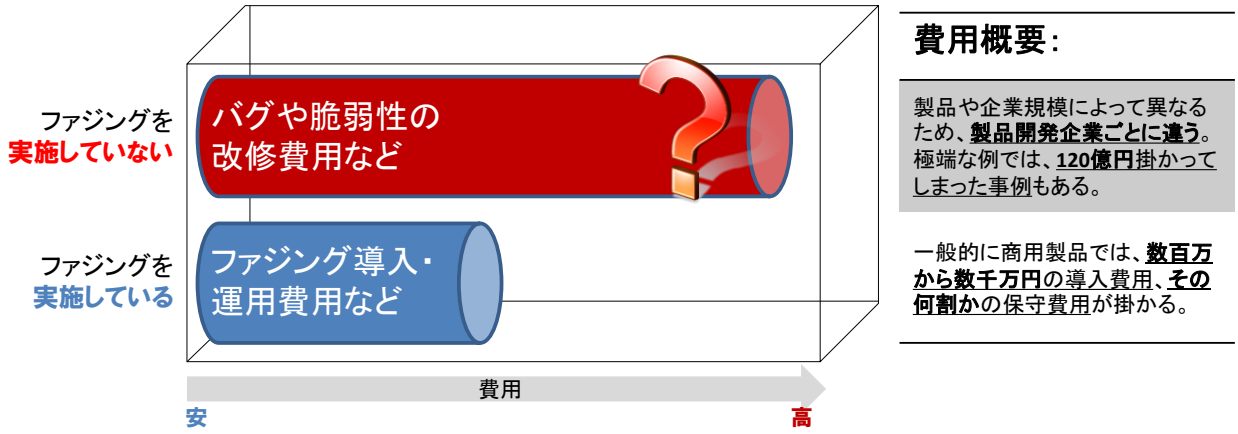


図 17 ファジング実施有無による問題発覚時の費用比較

■投資収益率（ROI: Return on Investment）の事例

ファジングツール導入・運用における投資収益率を算出した事例を紹介する。Codenomicon 社のレポート「The Total Economic Impact Of Codenomicon's Defensics Security Testing Suite」[14]では、ファジングツール「Codenomicon Defensics」導入顧客へのヒアリング結果を基に「Codenomicon Defensics」導入から 3 年間運用した場合の投資収益率を算出し、その結果をまとめている。

当該レポートによると、「Codenomicon Defensics」を導入して 3 年間運用した場合の投資収益率は 176%となり、約半年間で投資額を回収できると結論づけている（表 3）。

表 3 「Codenomicon Defensics」の投資収益率

投資収益率（ROI）	回収期間（Payback period）	総収益（Total benefits）	総費用（Total costs）	純現在価値（Net present value）
176%	0.5 年	2,583,997 ドル	937,759 ドル	1,646,237 ドル

13 ソニー株式会社：携帯電話回収に伴う費用について
http://www.sony.co.jp/SonyInfo/News/Press_Archive/200107/01-0706/

14 当該レポートはインターネット上に公開されていない。本書では、Codenomicon 社に許可をいただいたうえで、当該レポートの一部（表など）を和訳したうえで掲載している。

総収益、総費用の内訳は、それぞれ表 4、表 5 となる。総収益には、発見した問題の改修費用だけではなく、テストスクリプト開発費用やセキュリティテスト教育費用も削減対象となっている。総費用には、ファジングツールに関連する費用の他には、セキュリティテストそのものの実施費用(担当者の人件費など)が含まれる。

なお、表 4 と表 5 の数値と表 3 のものに差異があるが、これは当該レポートで採用している方法「Total Economic Impact(TEI)」によるものである。TEI では総収益、総費用におけるリスクを加味して投資収益率を算出しているために、この差異が生じている。TEI に言及することは本書の趣旨から外れるため、この差異については言及しない。

表 4 総収益の内訳

収益種目	初期収益	1 年目	2 年目	3 年目	合計
改修費用の削減(第 1 回目のテスト)		576,923	0	0	576,923
改修費用の削減(機能テストにおける発見)		288,462	288,462	288,462	865,385
改修費用の削減(出荷前テストにおける発見)		432,692	432,692	432,692	1,298,077
テストスクリプト開発費用の削減		173,077	173,077	173,077	519,231
セキュリティテスト教育費用の削減		9,231	9,231	9,231	27,692
総収益(リスク考慮していない)		1,480,385	903,462	903,462	3,287,308

単位:ドル

表 5 総費用の内訳

費用種目	初期費用	1 年目	2 年目	3 年目	合計
年間ライセンス費用	0	240,687	240,687	240,687	722,061
プロフェッショナルサービス費用	35,000	0	0	0	35,000
「Codenomicon Defensics」によるテスト実施費用	0	120,000	120,000	120,000	360,000
総費用(リスク考慮していない)	35,000	360,687	360,687	360,687	1,117,061

単位:ドル

■費用面からみたファジング導入の判断

上記の事例から、導入費用(初期費用や初年度費用)に数百万から数千万円が掛かってしまうが、ファジングを導入することで十分な費用対効果を見込めることが分かる。あとは導入・運用に掛かる費用を製品開発ライフサイクルの予算におさまるかが焦点になるだろう。

ファジングの導入・運用費用よりも製品開発ライフサイクルの予算が大きい場合には、品質を担保する意味でファジングの導入を検討してもよいだろう。特に「製品出荷後に問題が発生してその対応に掛かる費用」を試算したときに、その費用が非常に大きい場合[15]、品質を保証するテストとしてファジングを採用するとよいと考える。2.3.1 節で挙げたソフトウェア製品開発企業は、この場合に該当するために積極的にファジングを活用している。

だが、製品開発ライフサイクルの予算がファジングの導入・運用費用よりも小さい場合、ファジングの導入は難しいかもしれない。だが、その場合でもファジングを活用する方法があると考える。

15 「利用者が多く」、「社会インフラに近い」製品を開発している場合などを想定する。

4.2.2　テスト期間への影響

開発ライフサイクルにファジングを導入すると、新たなテストを 1 つ追加するため、最初は単純にテスト工数が増加する。だが何度もファジングを実施することで、「バグや脆弱性の発見数が増加しなくなるファジング工数」を確立することで、導入当初よりもテスト工数の増分は小さくなる（テスト工数が増加することに違いはない）。製品開発ライフサイクルにおける[テスト]工程で、このテスト工数の増分を検討する必要がある。

製品に対してファジングを実施する場合、時間を掛けてファジングを実施すれば、バグや脆弱性を発見できる可能性が高くなる。だが、[テスト]工程の限られたテスト期間で多大な時間を掛けることは現実的に難しいだろう。そこで、バグや脆弱性の発見数が増加しなくなる工数を見つけ、その工数だけファジングを実施することが効果的だと考える。

図 18 は Codenomicon 社が提供しているファジング代行サービス「Fuzz-o-Matic[16]」において 30 日間のファジングに掛けた日数（横軸）と発見したバグ等（縦軸）の関係を示している。Days（日数）が 6 日を過ぎたあたりで発見数が減少し始め、14 日以降では発見されていない。よって、図 18 では白点線部分の「9 日前後」がバグや脆弱性の発見数が増えなくなる工数となる。

また IPA の「脆弱性検出の普及活動」におけるファジング実績でも同様の傾向が見られた。ブロードバンドルータに対してファジングを実施した場合、ファジング期間の 1，2 日目でほとんどの脆弱性が発見できた（付録1を参照）。

製品企業ごとにファジングを実施しながら最適なテスト工数を見つけることで、テスト期間への影響を最小限におさえた効率的なファジングを実施できると考える。

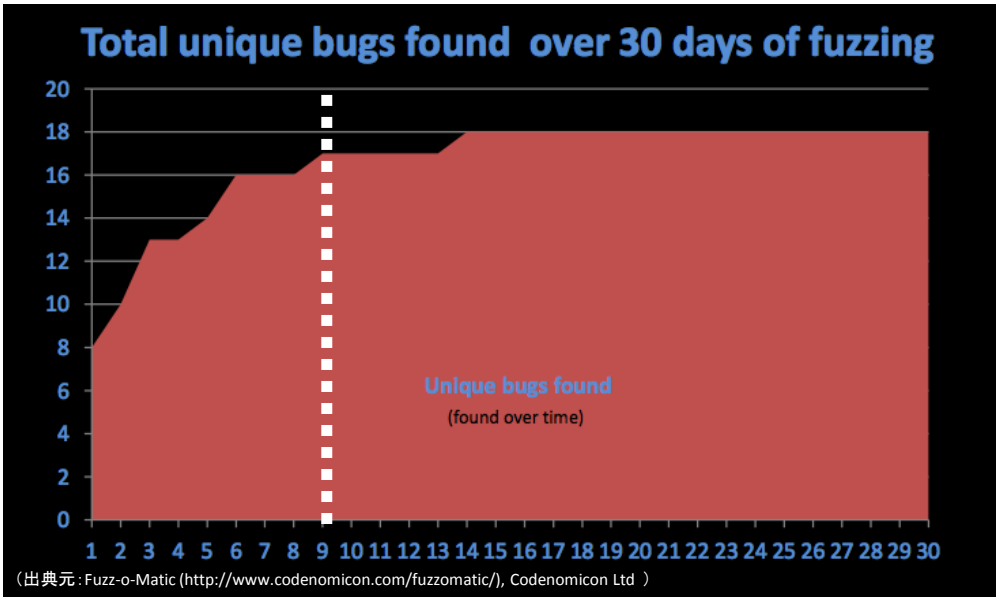


図 18 ファジングにおけるテスト日数と発見数の関係

16 http://www.codenomicon.com/fuzzomatic/

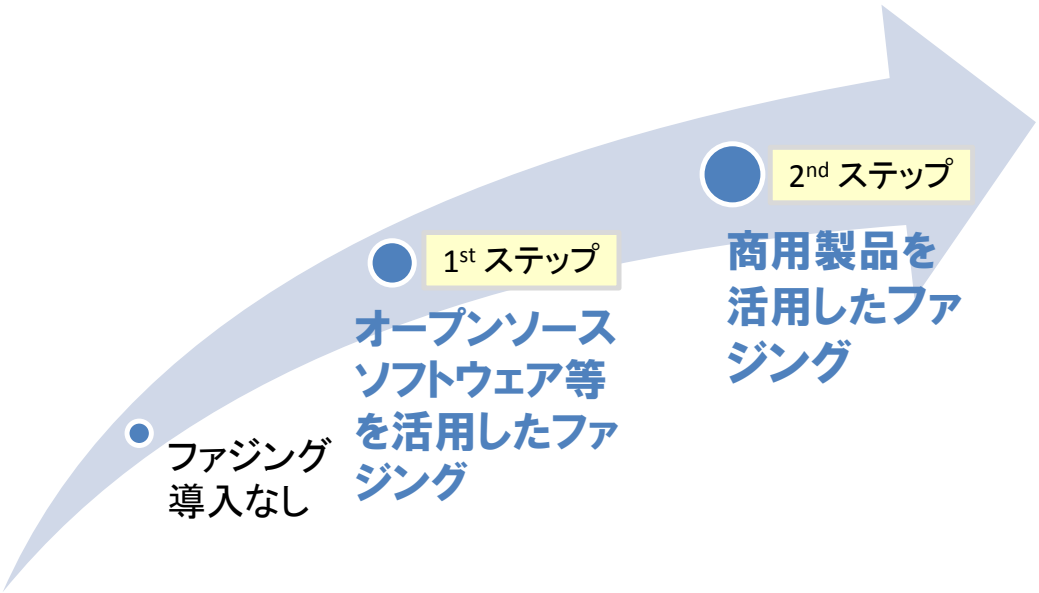
4.3 ファジング導入に向けたアプローチ

開発ライフサイクルでファジングを実施していない製品開発企業では、図 19 のようにファーストステップ（1st ステップ）、セカンドステップ（2nd ステップ）と分けてファジングの導入を検討することを提案したい。

まず 1st ステップとして、開発ライフサイクルに「オープンソースソフトウェア等を活用したファジング」を導入する。このファジングは費用・テスト期間への影響ともに小さいため、製品開発ライフサイクルへの導入の敷居が低いと考える。このファジングを実施することで、**インターネット上に公開されているツールで容易に発見できるバグや脆弱性を製品出荷前に発見できること**を期待できる。 IPA の「脆弱性検出の普及活動」を通じて、オープンソースソフトウェア等のファジングツールでも脆弱性を発見できる場合があった。まずはこういった脆弱性を製品出荷前に発見しよう。

「オープンソースソフトウェア等を活用したファジング」でファジングの効果を実感できたら、2nd ステップとして、商用製品の導入を検討してほしい。このファジングは費用・テスト期間への影響ともに大きい。だが、このファジングを実施することで、**製品出荷前により多くのバグや脆弱性を発見でき、製品の安全性を高めること**を期待できる。4.5 節で説明するが、IPA の活動実績に基づくと商用製品のファジングツールは高価であるが、オープンソースソフトウェア等のファジングツールよりもバグや脆弱性を多く発見できた。

4.4 節以降で、「オープンソースソフトウェア等を活用したファジング」、「商用製品を活用したファジング」を順に説明してきたい。



	1stステップ	2ndステップ
①費用	【安い】	【高い】
②テスト期間	影響【小】	影響【大】

図 19 ファジング導入に向けたアプローチ（IPA の考えに基づく）

4.4 [1st ステップ]オープンソースソフトウェア等を活用したファジング

ファジング活用の 1st ステップとして、まずオープンソースソフトウェア等を活用したファジングを実施しよう。IPA は[テスト]工程、[実装]工程ごとのファジング活用方法を図 20 のように考えている。4.4.1 節、4.4.2 節でそれぞれの活用方法を説明したい。

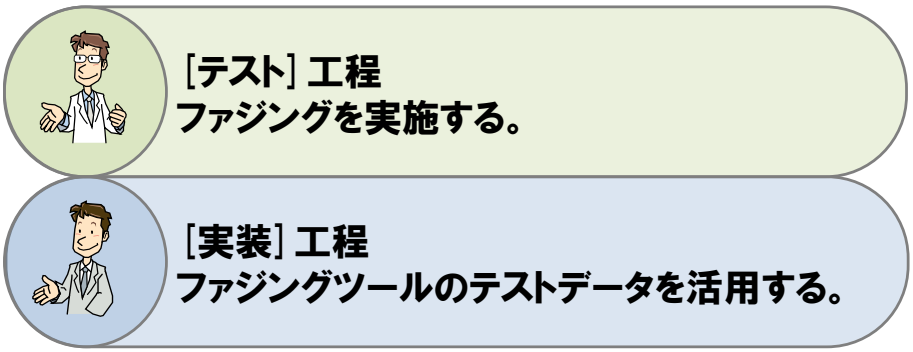


図 20 オープンソースソフトウェア等を活用したファジング

4.4.1 [テスト]工程におけるファジングの活用

[テスト]工程では、製品が備えている機能に対応するオープンソースソフトウェアを選別して、それらを使ってファジングを実施するのがよいだろう。

オープンソースソフトウェア等のファジングツールには、特定の機能へのファジングに特化したツールが数多く存在している。「脆弱性検出の普及活動」で確認した限りでは[17]、表 6 のファジングツールを確認した。ブロードバンドルータへのファジングでは、オープンソースソフトウェアのファジングツールを使うと、すぐに脆弱性を発見できた場合があった。このことから、表 6 のファジングツールなどを活用することで第三者が容易に発見できてしまう脆弱性を[テスト]工程で発見することを期待できる。

表 6 オープンソースソフトウェア等のファジングツール

テスト対象機能	対応するファジングツール例
TCP/IP 通信機能	● ISIC（IP Stack Integrity Checker） ● fuzzball2
アプリケーション通信機能 （HTTP 通信や FTP 通信など）	● Taof（The art of fuzzing） ● Peach
ファイル処理機能 （画像ファイルなど）	● SDL MiniFuzz File Fuzzer ● BFF（CERT Basic Fuzzing Framework）
ウェブブラウジング機能	● cross fuzz ● DOM-Hanoi ● Hamachi ● CSSDIE ● <CANVAS> fuzzer ● Browser Fuzzer 3

17 「ファジング活用の手引き」（http://www.ipa.go.jp/security/vuln/fuzzing.html）の「付録 A. ファジングツールの紹介」でもオープンソースソフトウェア等のファジングツールを紹介している。

4.4.2 [実装]工程におけるファジングの活用

[実装]工程では、開発時の動作テスト（開発したプログラムが正常に動作するか、適当な値をプログラムに入力する動作確認など）でオープンソースソフトウェア等のファジングツールにおけるテストデータを活用できるだろう。

オープンソースソフトウェア等のファジングツールには、「脆弱性を悪用する攻撃などに使われる値」が登録されている。例えば、表 7 のような値がある。製品にこういった脆弱性があると、最悪の場合製品上での不正なコード実行につながってしまう。[実装]工程におけるテストでこのような値を入力値として使うだけでも脆弱性の発見を期待できる（より具体的な値については「ファジング実践資料[18]」の 1.5 節、2.6 節で紹介している）。

この活用方法では、[実装]工程ですでに実施しているテストにテスト値を上乗せするだけであり作業項目自体が増加しない。このことから、現在の開発体制に導入しやすいファジングの活用方法と言える。

表 7 脆弱性を悪用する攻撃などに使われる値の例

脆弱性を悪用する攻撃などに使われる値	値の説明
AAA・・・（「A」を 65535 回繰り返す）	開発中の製品に「バッファオーバーフロー」という脆弱性があった場合、こういった文字列を入力することで、製品が異常終了してしまう。
%s%s%s・・・（「%s」を 30 回繰り返す）	開発中の製品に「書式文字列の問題」という脆弱性があった場合、こういった文字列を入力することで、製品が異常終了してしまう。
65536	開発中の製品に「整数オーバーフロー」という脆弱性があった場合、こういった数値を入力することで、製品が異常終了してしまう。

[18] IPA：別冊：「ファジング実践資料」
http://www.ipa.go.jp/security/vuln/documents/fuzzing-tool.pdf

4.5 [2nd ステップ]商用製品を活用したファジング

「オープンソースソフトウェア等を活用したファジング」で、実際にバグや脆弱性を発見しその効果を実感できたら、続いて「商用製品を活用したファジング」の実践を検討してほしい。それは「商用製品」の方がより多くのバグや脆弱性を発見できるからだ。

IPA における「脆弱性検出の普及活動」では 2011 年 8 月から 2012 年 6 月末日にかけて、計 17 台の組込み機器に対してファジングを実施した。この活動の結果、24 件の脆弱性を検出した。このうちオープンソースソフトウェア等のファジングで発見した脆弱性は 4 件(12.5%)にとどまり、残りの 20 件は商用製品のファジングツールで発見している。

オープンソースソフトウェア等のファジングツールと商用製品のものを比較した場合、「ファジングを実施する箇所」と「ファジングにおけるテストデータ」の網羅性に違いがあり、脆弱性の発見数に差がでる。4.5.1 節以降、ファジングツールによって発見できる脆弱性が異なる理由を説明していきたい。

4.5.1 ファジングツールにおけるテストデータ

ファジングでバグや脆弱性を発見できるかどうかは、使用するファジングツールのテストデータによって決まる。ファジングツールのテストデータは、対象製品の「ファジングの対象とする機能(ファジング対象機能)」ごとに用意される。このテストデータは、「ファジング対象機能」ごとにその機能の「どの範囲」に「どんな値」を送るか、という点で決まる(図 21)。このことから、「製品のどの機能」の「どの範囲」を「どんな値」でファジングを実施するか、ファジングツール毎の設計思想によって違いがでてくる。

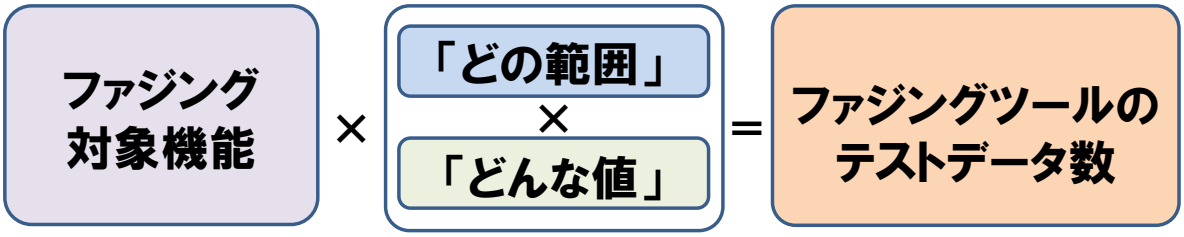


図 21 ファジングツールのテストデータ

商用製品「Codenomicon Defensics[19]」と「FFR Raven[20]」、オープンソースソフトウェア「Taof[21]」で、「IPv4 処理機能」、「HTTP 処理機能」に対してファジングを実施する場合を例に、それぞれのファジングツールが「どの範囲」を「どんな値」でファジングを実施するか例示しよう。

---

19 Codenomicon 社：Codenomicon Defensics
http://www.codenomicon.com/defensics/

20 株式会社フォティーンフォティ技術研究所：FFR Raven
http://www.fourteenforty.jp/products/raven/

21 Taof - The art of fuzzing
http://sourceforge.net/projects/taof/

4.5.2 「IPv4 処理機能」に対するファジングのテストデータ例

「IPv4 処理機能」に対するファジングを実施する場合、テストデータは IPv4 パケットとなる（図 22）。IPv4 パケットをテストデータとしてみる場合、ファジング範囲として「IPv4 ヘッダ部分」と「IPv4 データ部分」がある。ファジングツールはこれらの中からファジングする範囲を選び、テストデータを埋め込んでいく。

商用製品「Codenomicon Defensics」、「FFR Raven」、オープンソースソフトウェア「Taof」が、IPv4 パケットの「どの範囲」を選び、「どんな値」を埋め込むかみてみよう。

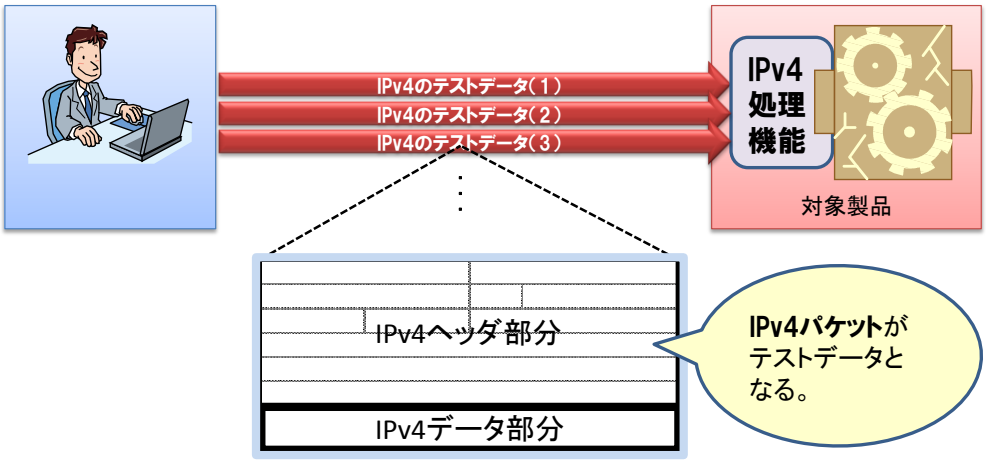


図 22 「IPv4 処理機能」に対するファジングのテストデータ

3 つのファジングツールが IPv4 パケットの「どの範囲」を選ぶか、図 23 にまとめた。図 23 の青色部分がそれぞれのファジングツールがファジングする範囲となる。

「Codenomicon Defensics」は「IPv4 ヘッダ」と「IPv4 データ」のすべてを対象にしているが、「FFR Raven」は「IPv4 ヘッダ」の一部分のみにとどまることが分かるだろう。さらに、「Taof」ではそもそも「IPv4 処理機能」に対してファジングを実施できない。

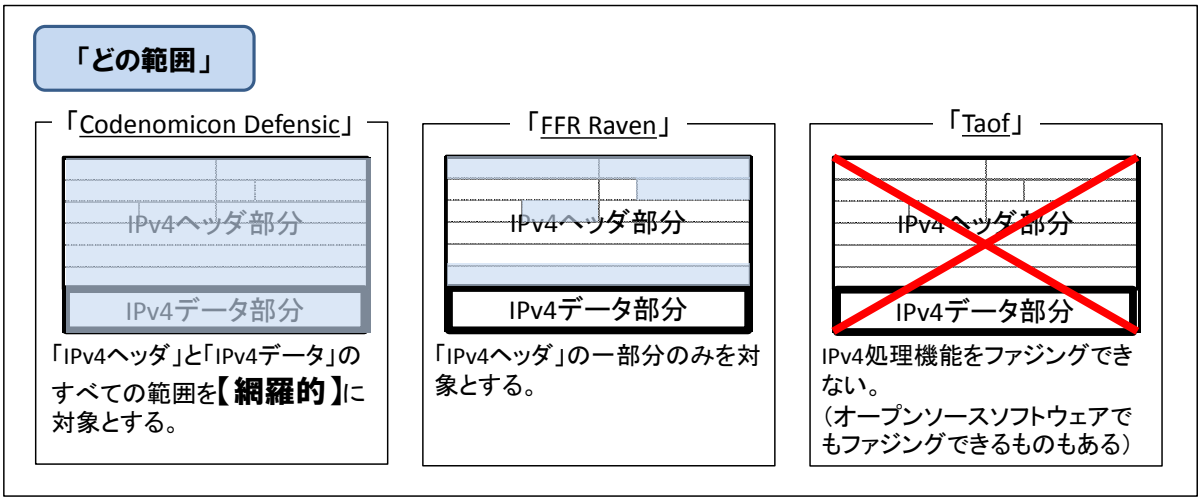


図 23 IPv4 パケットの「どの範囲」を選ぶか

では、図 23 で図示した範囲に対して、「どんな値」を設定して実際にファジングを実施するか、それを図 24 にまとめた(「Taof」では「IPv4 処理機能」に対してファジングを実施できないため割愛した)。

「IPv4 ヘッダ」の各フィールドは値の大きさが決まっている。そのため、設定される値としては「取り得るほぼすべての値」(図 24 の赤色部分)を選択するか、「問題が起きそうな値」(図 24 の橙色部分)を選択するかの 2 通りに分かれる。

「Codenomicon Defensics」は網羅的な範囲をファジングする分だけ「問題が起きそうな値」のみを指定する。一方で、「FFR Raven」は一部分のみしかファジングしない分だけ「取り得るほぼすべての値」を設定する。

以上のことから、「IPv4 処理機能」に商用製品「Codenomicon Defensics」と「FFR Raven」でファジングを実施した場合、それぞれの結果に違いがでてくる。

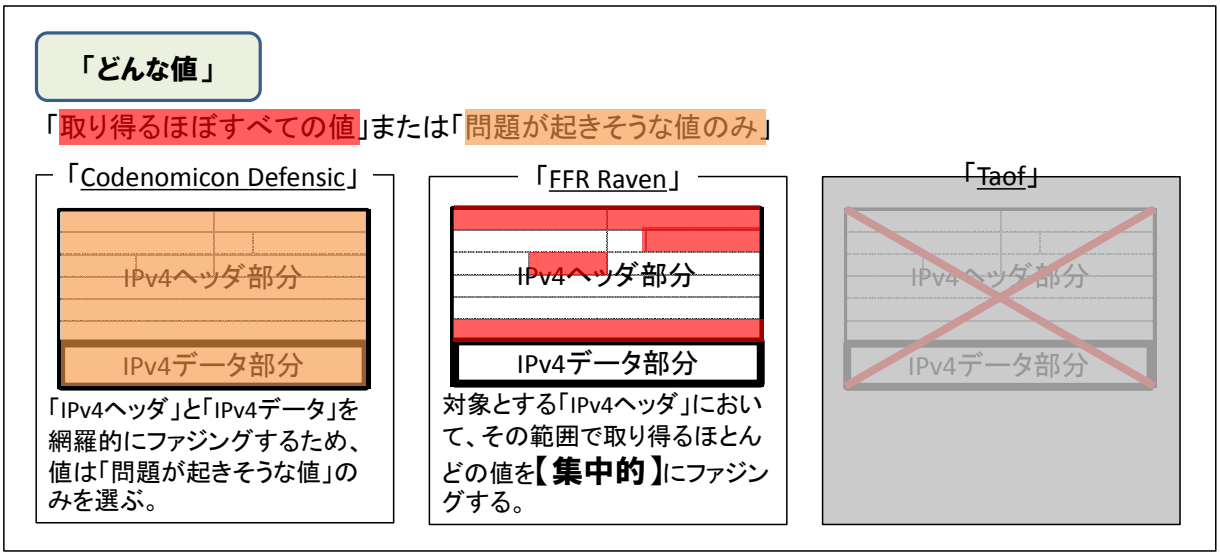


図 24 選んだ範囲に「どんな値」を設定するか

4.5.3 「HTTP 処理機能」に対するファジングのテストデータ例

続いて、「HTTP 処理機能」に対するファジングのテストデータ例を見てみよう。

「HTTP 処理機能」に対するファジングを実施する場合、テストデータは HTTP リクエストとなる（図 25）。HTTP リクエストをテストデータとしてみる場合、ファジング範囲として「HTTP ヘッダ」と「HTTP ボディ」がある。ファジングツールはこれらの中からファジングする範囲を選び、テストデータを埋め込んでいく。

商用製品「Codenomicon Defensics」、「FFR Raven」、オープンソースソフトウェア「Taof」が、HTTP リクエストの「どの範囲」を選び、「どんな値」を埋め込むかみてみよう。

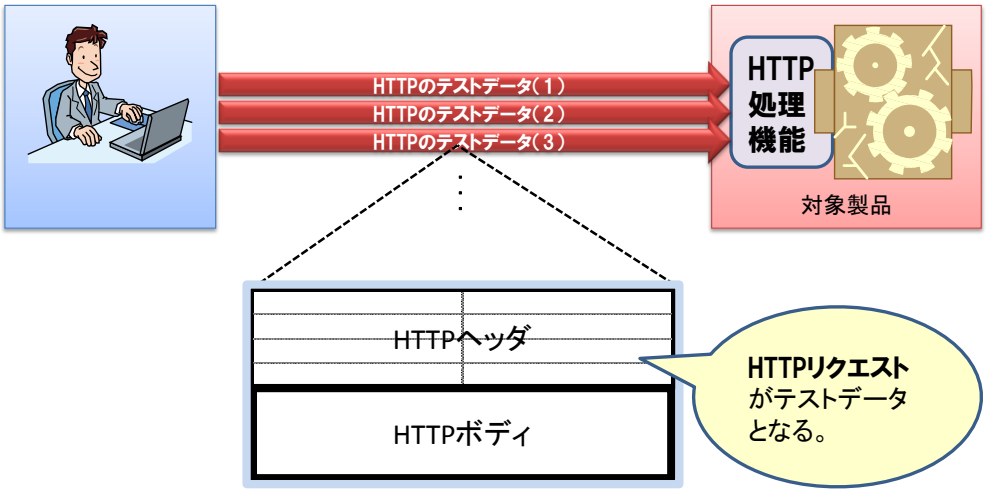


図 25 「HTTP 処理機能」に対するファジングのテストデータ

3 つのファジングツールが HTTP リクエストの「どの範囲」を選ぶか、図 26 にまとめた。図 26 の青色部分がそれぞれのファジングツールがファジングする範囲となる。

「Codenomicon Defensics」、「FFR Raven」ともに、利用者が用意したサンプル HTTP リクエストを基づき「HTTP ヘッダ」と「HTTP ボディ」を対象とする。だが一方で、「Taof」は利用者がどこをファジングするかを決めなければいけない。このことから、「Taof」の場合、ファジングにより脆弱性を発見できるか否かが利用者次第となってしまう。

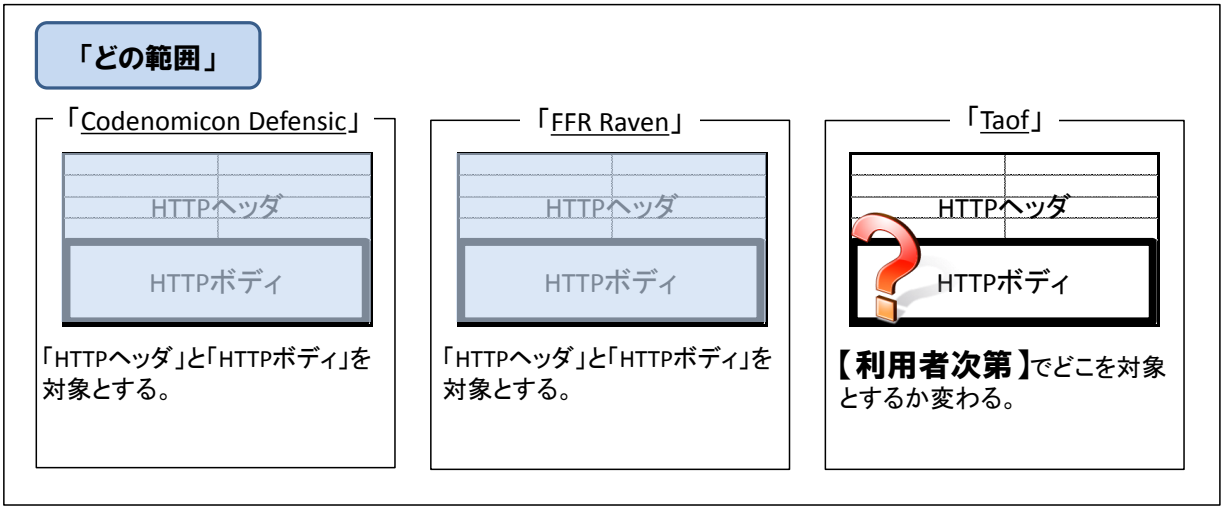


図 26 HTTP リクエストの「どの範囲」を選ぶか

では、図 26 で図示した範囲に対して、「どんな値」を設定して実際にファジングを実施するか、それを図 27 にまとめた。

「HTTP ヘッダ」と「HTTP ボディ」の各フィールドは値の大きさが決まっていないため、値を自由に設定できる。そのため、ファジングツールは基本的に「問題が起きそうな値」（図 27 の橙色部分）を選択する。ファジング対象の範囲にどんな値を設定するかは、ファジングツール開発者のノウハウに依存する。

このような場合、「Codenomicon Defensics」、「FFR Raven」、「Taof」で該当部分の脆弱性を発見できるか否かはツール次第と言える。

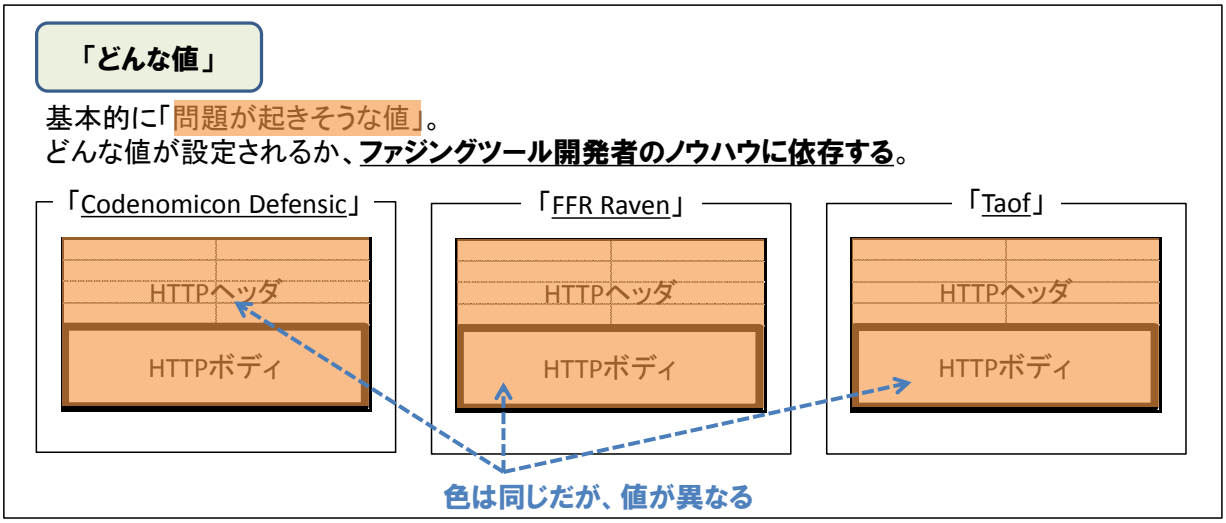


図 27 選んだ範囲に「どんな値」を設定するか

4.5.4　ファジングツールの特徴

4.5.2 節、4.5.3 節から、同じ機能に対するファジングであっても、商用製品「Codenomicon Defensics」と「FFR Raven」、オープンソースソフトウェア「Taof」を比べた場合、その機能の「どの範囲」を「どんな値」でファジングを実施するかが異なる。4.5.2 節、4.5.3 節の結果をふまえると、「Codenomicon Defensics」、「FFR Raven」、「Taof」の特徴は、図 28 となる。

このように「どの機能」に対して「どの範囲」を「どんな値」でファジングを実施するかによって、ファジングツールに特徴がでる。この特徴の違いから、ファジングツールによって発見できる脆弱性が異なる。図 28 からオープンソースソフトウェアよりも商用製品の方がより【網羅的】、または【集中的】にファジングを実施できることが分かる。

一方、オープンソースソフトウェアは商用製品よりもファジングの対象となる機能が少ない場合があり、バグや脆弱性を発見できるかが利用者次第となってしまう。だが、4.4.1 節で触れたようにオープンソースソフトウェアのファジングツールは多く存在しているため、製品にあわせてこれらのファジングツールもうまく活用したい。

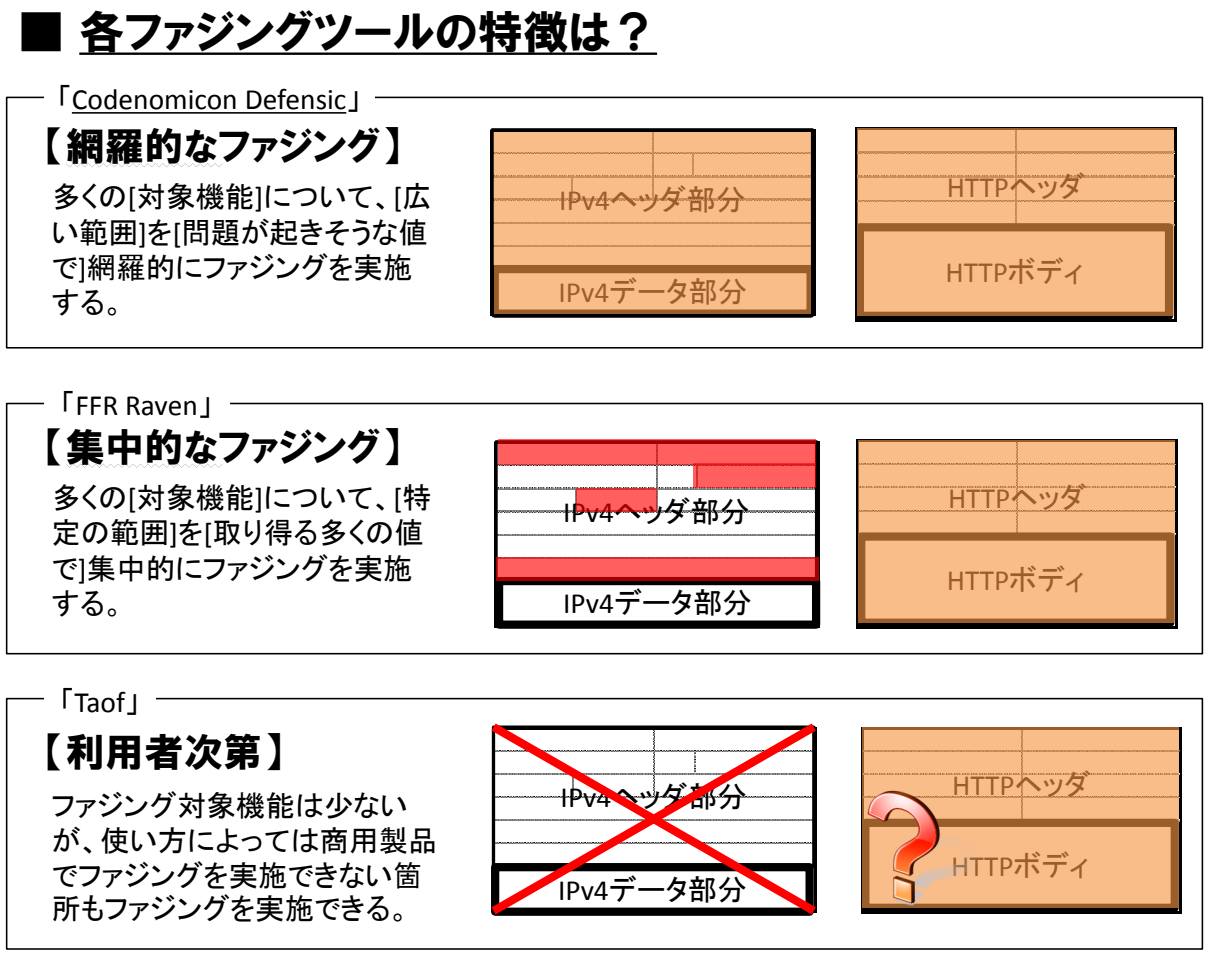


図 28 ファジングツールの特徴

5　まとめ

IPA では「脆弱性検出の普及活動」を通じて 2011 年 8 月から約 1 年間、様々な組込み機器にファジングを実施してきた。この結果、以下のことが明らかになった:

- **実際に出荷されている製品からファジングで脆弱性を発見できた。**
  ファジングは有効なテストの一つである。
- **ファジングツールによって発見できる脆弱性に差異がある。**
  テストの目的(とにかく網羅的にテストしたい、実施しているテストで足りなそうな箇所を集中的に検査したいなど)に応じたファジングツールを活用することで、効果的なセキュリティテストを実施できる。

ファジングを導入していない製品開発企業には、まずはオープンソースソフトウェア等を活用したファジングを是非実践してみていただきたい。商用製品を購入して本格的にファジングを実践する場合、確かに費用が掛かる。だが、オープンソースソフトウェアよりも利用者に依存することなく、製品出荷前により多くのバグや脆弱性を機械的に発見でき、製品の品質確保につながると考える

本レポートがファジングを理解、活用する一助となれば幸いである。

6　今後の IPA の取り組み

本書を公開した後も、引き続き「脆弱性検出の普及活動」を継続する。今後は表 8 の①から③の活動を予定している。また、本活動で発見した脆弱性を製品開発者に報告すると共に、本活動で得られた知見を公開していく予定である。IPA の活動で得られた知見などが活用され、製品出荷前により多くのバグや脆弱性が解消されることを期待する。

表 8 「脆弱性検出の普及活動」今後の予定

活動	活動内容
①	**日本製および海外製のインターネットにつながるデジタルテレビに対するファジング** 国内外問わずインターネットにつながるデジタルテレビを調達して、実際にそれらに対してファジングを実施する予定である。このファジングで得られた知見を製品開発者と共有して、デジタルテレビ出荷前にバグや脆弱性を発見できる製品開発に寄与していく。
②	**制御システムセキュリティセンター(CSSC)[22]との連携** CSSC と連携して、「脆弱性検出の普及活動」で培ったノウハウを活用しながら、制御システムに対してファジングを実施していく予定である。この CSSC との連携を通じて、制御システムにおける品質向上に寄与していく。
③	**脆弱性検出対象の拡充** 現在組込み機器を中心にファジングを実施することでファジングの有効性を実証しているが、今後は組込み機器以外にもソフトウェア製品やウェブアプリケーションにもその範囲を拡大していく予定である。

[22] 技術研究組合 制御システムセキュリティセンター（CSSC）
http://css-center.or.jp/

付録１：「脆弱性検出の普及活動」における脆弱性発見までの期間とテスト件数

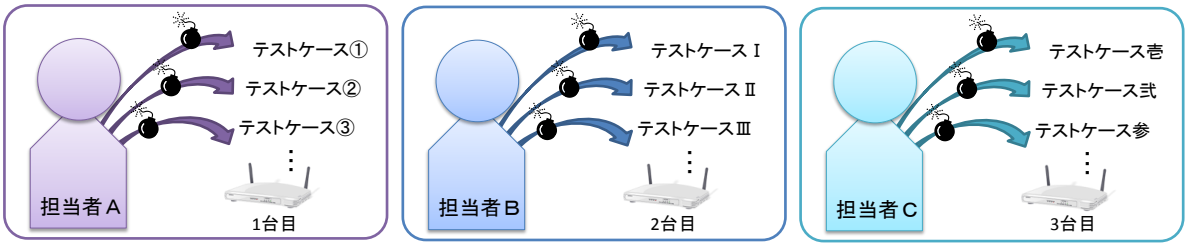
IPA の「脆弱性検出の普及活動」では、ブロードバンドルータへのファジングで脆弱性を発見した（表 9）。本付録では、脆弱性を検出したブロードバンドルータ A, D, F に対するファジングにおいて、「脆弱性発見までの期間」と「（ファジングツールによる）総テスト件数」を紹介する。

表 9 「脆弱性検出の普及活動」におけるブロードバンドルータでの検出実績

対象機器	脆弱性発見数（*1）	延べ 日数（*2）
ブロードバンドルーターA	2 件	9日
ブロードバンドルーターB	0 件	7日
ブロードバンドルーターC	0 件	9日
ブロードバンドルーターD	3 件	6日
ブロードバンドルーターE	0 件	5日
ブロードバンドルーターF	1 件	9日

（*1）：LAN側でこれらの脆弱性を検出しており、インターネットのWAN側でこれらの脆弱性による事象を再現できませんでした。
（*2）：この延べ日数は単純にファジングを実践した時間だけではなく、検査パターンの特定などの時間も含みます。またファジング担当者はこの作業に専任したわけではありません。

なお、本付録で紹介するファジング結果は、図 29 の体制とツールで実施した結果である。本付録で紹介するファジング結果は 3 名で並列にファジングを実施した結果であるため、1 名でファジングを実施する場合、単純に計算すると **3 倍**の時間が掛かることに留意していただきたい。



ツール名	商用製品／オープンソースソフトウェア
Codenomicon Defensics	商用製品
Peach	オープンソースソフトウェア
Taof	オープンソースソフトウェア

図 29 IPA におけるファジング体制と使用ツール

■結果分析

ブロードバンドルータ A, D, F に対するファジングの「脆弱性発見までの期間」と「(ファジングツールによる)総テスト件数」の分析から次のことが分かる。

- ファジング期間全体のうち、1 日目または 2 日目で脆弱性を発見できた。
- ファジング期間全体の 70-80%の期間を「Codenomicon Defensics」の「TCP for IPv4 Server Test Suite」によるファジングに費やしていた。このファジングの効率化をすすめると、ファジング期間全体の短縮が見込める。

■ブロードバンドルータ A

ブロードバンドルータ A におけるファジングのテスト件数と脆弱性発見数を、図 30 に示す。ブロードバンドルータ A では IPA が発見した 2 つの脆弱性を発見するまでに 2 日間掛かっている。

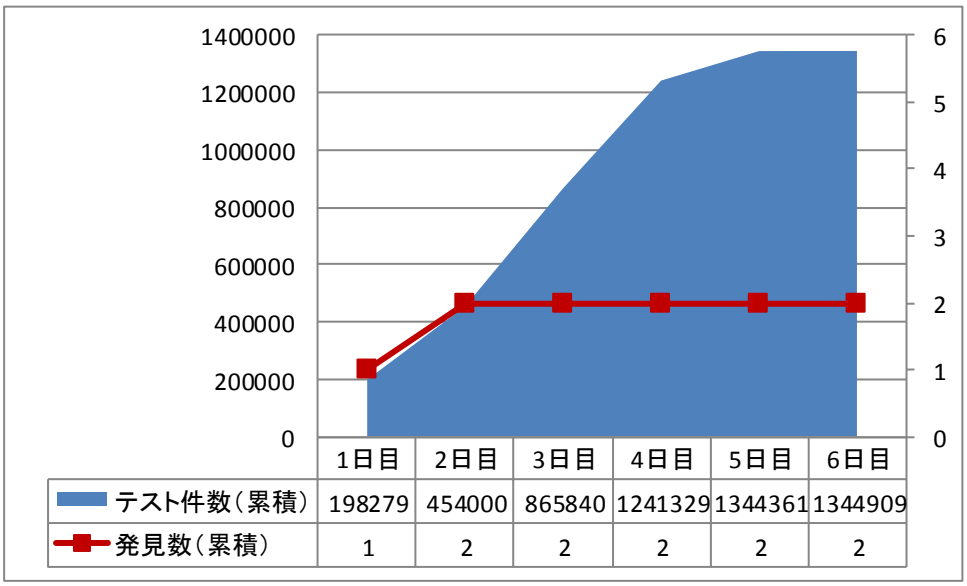


図 30 ブロードバンドルータ A におけるファジングのテスト件数と脆弱性発見数

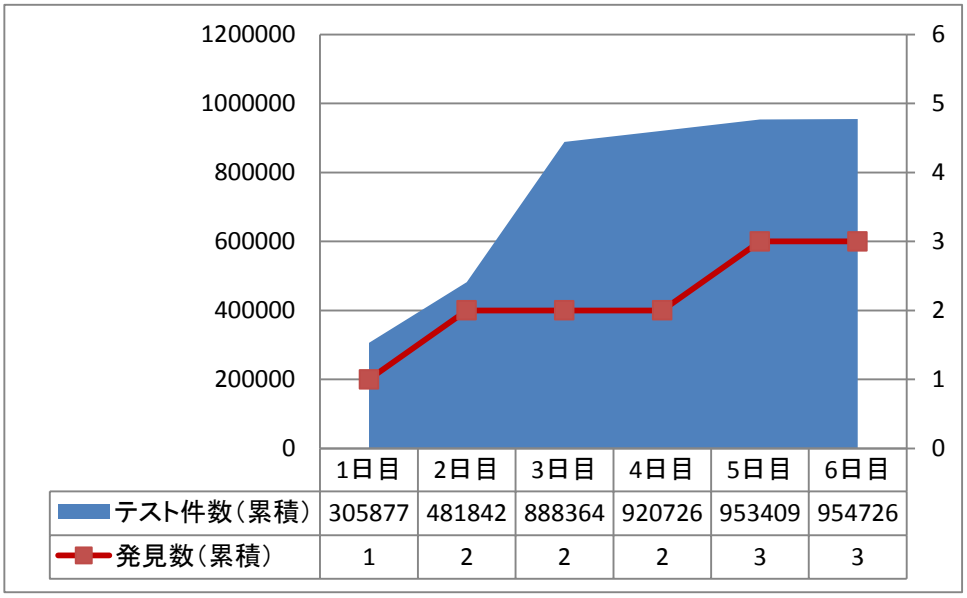
ブロードバンドルータ A におけるファジングのツール毎のテスト件数とテスト時間の内訳を表 10 にまとめた。総テスト時間の 75%を「Codenomicon Defensics」の「TCP for IPv4 Server Test Suite」を占めている。

表 10 ブロードバンドルータ A におけるファジングのテスト件数とテスト時間

		テスト件数	テスト時間
Codenomicon		1,227,310	61:31:25
	IPv4 Test Suite	161,797	3:57:21
	ICMPv4 Test Suite	140,477	0:46:32
	DHCPv4 Test Tool	27,808	8:19:21
	ARP Server Test Suite	748,142	3:23:15
	TCP for IPv4 Server Test Suite	149,086	45:04:56
Peach		116,918	7:29:34
Taof		681	0:05:07
	合計	1,344,909	69:06:06

■ブロードバンドルータ D

ブロードバンドルータ D におけるファジングのテスト件数と脆弱性発見数を、図 31 に示す。ブロードバンドルータ D では IPA が発見した 3 つの脆弱性を発見するまでに 5 日間掛かっている。



	1日目	2日目	3日目	4日目	5日目	6日目
テスト件数(累積)	305877	481842	888364	920726	953409	954726
発見数(累積)	1	2	2	2	3	3

図 31 ブロードバンドルータ D におけるファジングのテスト件数と脆弱性発見数

ブロードバンドルータ D におけるファジングのツール毎のテスト件数とテスト時間の内訳を表 11 にまとめた。総テスト時間の 81%を「Codenomicon Defensics」の「TCP for IPv4 Server Test Suite」を占めている。

表 11 ブロードバンドルータ D におけるファジングのテスト件数とテスト時間

		テスト件数	テスト時間
Codenomicon		802,658	90:28:42
	IPv4 Test Suite	148,619	2:31:41
	ICMPv4 Test Suite	140,477	2:04:03
	DHCPv4 Test Tool	6,313	3:15:25
	ARP Server Test Suite	374,671	4:55:05
	TCP for IPv4 Server Test Suite	132,578	77:42:28
Peach		149,120	4:23:00
Taof		2,948	1:02:00
	合計	954,726	95:53:42

■ブロードバンドルータ F

ブロードバンドルータFにおけるファジングのテスト件数と脆弱性発見数を、図 32に示す。ブロードバンドルータ F では IPA が発見した1つの脆弱性をファジング開始初日に発見している。

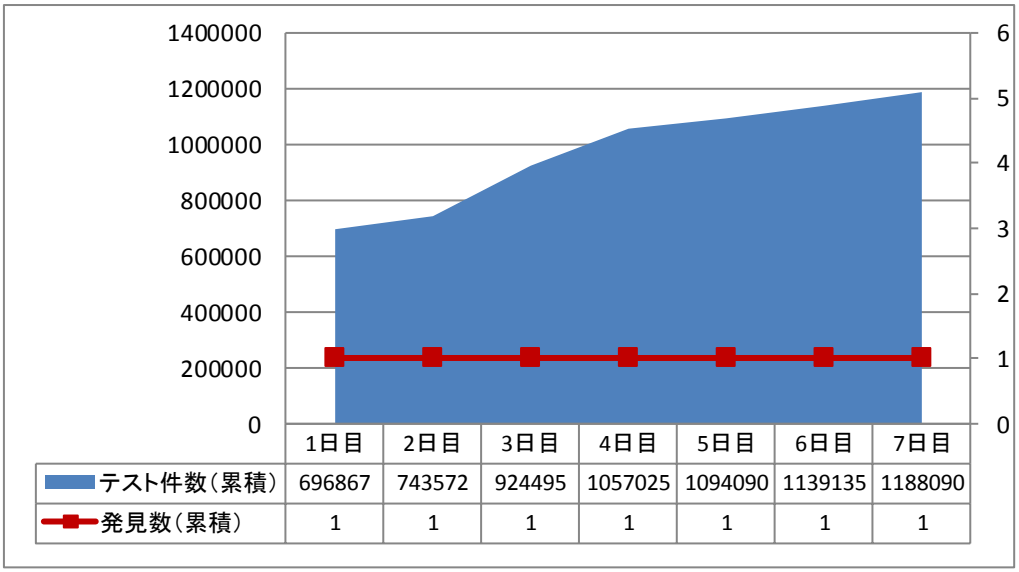


図 32 ブロードバンドルータ F におけるファジングのテスト件数と脆弱性発見数

ブロードバンドルータ F におけるファジングのツール毎のテスト件数とテスト時間の内訳を表 12 にまとめた。総テスト時間の 69%を「Codenomicon Defensics」の「TCP for IPv4 Server Test Suite」を占めている。

表 12 ブロードバンドルータ F におけるファジングのテスト件数とテスト時間

		テスト件数	テスト時間
Codenomicon		1,008,142	83:10:31
	IPv4 Test Suite	150,557	5:47:59
	ICMPv4 Test Suite	134,509	1:59:02
	DHCPv4 Test Tool	600	0:37:23
	ARP Server Test Suite	411,201	5:23:24
	TCP for IPv4 Server Test Suite	311,275	69:22:43
Peach		179,276	16:42:44
Taof		672	0:02:00
	合計	1,188,090	99:55:15

以上