平成 28 年度修士論文

シンスライシングを用いた PHP アプリの設定値参照ミス検知手法の提案

電気通信大学 大学院情報システム学研究科
社会知能情報学専攻

学籍番号 : 1451035
氏名 : 依田 みなみ

主任指導教員 : 大須賀 昭彦 教授
指導教員 : 田原 康之 准教授
指導教員 : 植野 真臣 教授

提出年月日 : 平成 29 年 1 月 26 日 (木)

概要

PHP5 で開発されたアプリケーションでは，設定値の参照ミスによってアプリケーションに不具合が発生することがある．本研究は PHP5 アプリケーションを対象とした設定値の参照ミスを検知する手法を提案する．また実際に提案手法を実現した，設定値参照ミス検知ツール『Mis.Config』を開発した．

PHP5 で開発されたアプリケーションは，設定ファイル内の設定値がプログラム内で参照されるとき，設定ファイル内で定義された型とは異なる型を想定して参照されることがあり，これによってアプリケーションが誤動作を引き起こすバグが報告されている．これまでも，ソフトウェアの設定ミスが引き起こすバグの検知については様々な手法が提案されてきたが，本研究が対象とする設定値の参照ミスに起因するバグを対象とした手法は提案されていない．

提案手法では，設定値の参照ミス，つまり設定値参照時の型不一致を検知するため，設定ファイル内の設定値の型と PHP プログラム内の設定値の型を照合し，不一致だった場合にその旨を報告する．PHP5 は変数の型を推論しないため，本手法は独自のルールに則って設定値の型を特定する．また本手法では，PHP プログラム内で設定値を格納しているローカル変数も設定値とみなし，型不一致検査の対象とした．

更に本研究では，提案手法を実現した型不一致検知ツール『Mis.Config』を開発した．Mis.Config は，型不一致を検知したい PHP アプリケーションを解析し，結果として型不一致が生じている設定値の有無，型不一致が発生した設定値名と該当プログラム文を報告する．ツールを実現するにあたり，PHP プログラムの解析にコントロールフローグラフとシンスライシングを用いた．

Mis.Config の精度実験として，実際に公開されている PHP アプリケーションの設定値を変更・追加し，人為的にプログラム内と設定ファイル間で，設定値の型不一致を発生させた．そして，Mis.Config が型の不一致を検知できるかの精度実験を行った．結果，本研究が対象とした整数型・文字列型・論理型の，三種類の型間で発生した型不一致の再現率・適合率は共に 100% であった．全体の精度は，適合率 100%, 再現率 78%, F 値 85% であった．

目次

第 1 章　序論 1
1.1　背景 1
1.2　研究目的と概要 2
1.3　本論文の構成 3

第 2 章　実アプリの設定値参照ミス関するバグ報告 4
2.1　Joomla! 1.5 4
2.2　WebChess 5
2.3　参照ミスが発生しうる設定値の例 7

第 3 章　関連研究 9

第 4 章　背景知識 11
4.1　PHP（Hypertext Preprocessor） 11
4.2　スライシング 12
4.2.1　フルスライシング 12
4.2.2　シンスライシング 12
4.2.3　フルスライシングとシンスライシングの解析結果の比較 12
4.3　コントロールフローグラフ 15
4.4　適合率・再現率・F 値 15

第 5 章　提案手法 17
5.1　提案手法の概要 17
5.2　設定値の型を特定する 17
5.3　設定値を格納しているローカル変数の特定 19
5.4　設定値の型一致検査 19

5.5 解析結果の表示 21

第 6 章 Mis.Config：提案手法の実装 22
6.1 Mis.Config 概要 22
6.2 開発環境 22
6.3 コントロールフローグラフの実装 23
6.4 設定値の型を特定する 24
6.4.1 設定値の型特定に用いる正規表現リスト 24
6.5 設定値を格納しているローカル変数の特定 25
6.6 設定値の型一致検査 26
6.7 解析結果の表示 27

第 7 章 精度評価実験 28
7.1 精度実験概要 28
7.1.1 実験で変更・追加した設定値 29
7.2 精度実験の結果と考察 31
7.2.1 整数型に変更された設定値 31
7.2.2 論理型に変更された設定値 32
7.2.3 文字列型に変更された設定値 33
7.2.4 その他の値に変更された設定値 34
7.2.5 Joomla!1.5 で報告された型不一致ミスの再現 35

第 8 章 議論と今後の課題 38

第 9 章 まとめ 40

参考文献 42

付録 A Mis.Config のプログラム 45

謝辞 50

研究業績 51

図目次

2.1 WebChess のチェスゲームインストール画面 7

4.1 シンスライシングとフルスライシングの比較 13
4.2 コントロールフローグラフ 14

5.1 提案手法の流れ 18

6.1 コントロールフローグラフのノード情報 23
6.2 設定値型特定の解析画面 25
6.3 本手法の結果表示画面 27

7.1 Joomla! 1.5 で報告されたバグの再現画面 37

表目次

4.1 ツールの評価指標 15

5.1 PHP アプリケーション設定ファイルの設定値の型特徴 18

6.1 設定ファイル内の設定値の型特定に用いる正規表現 24

7.1 整数型に変更された設定値 29
7.2 論理型に変更された設定値 29
7.3 文字列型に変更された設定値 30
7.4 その他の型に変更された設定値 30
7.5 精度実験 30

第 1 章

序論

1.1 背景

PHP アプリケーション開発において，設定ファイルは多く活用されている．設定ファイルとは，アプリケーション内で用いる固定値をまとめたファイルである．固定値（設定値）とは，アプリケーションの動作に必要な情報を持ち，またアプリケーションによって値が変更されることがない値を指す．例えば，データベース接続に用いるデータベース名・ユーザー名・パスワードの情報は，アプリケーションの動作に必要な情報かつ，アプリケーションの動作中に値が変更されることはない．

設定ファイルに設定値をまとめることによって，アプリケーションの設定情報を一覧でみることができる．設定値の保存場所が分散していないので，変更・修正も容易である．

可読性の高いアプリケーション開発の重要性はこれまで多く議論されていきている．Buse ら [1] は，ソフトウェアの可読性とその品質の関連性を調査した．調査の結果，可読性の低いソフトウェア内でバグが検出される可能性は，可読性の高いソフトウェアよりも 150% 以上であることがわかっている．よって，設定ファイルを作成し設定値を一箇所で管理することで，情報の可読性を向上させることは，アプリケーションの保守管理に役立つといえる．

一方で，設定ファイルの利用はデメリットもある．設定ファイルに保存されていた設定値は，プログラム内の適当な箇所で参照される．プログラムから設定値を参照する際，ヒューマンエラーなどによって，参照する設定値を間違えてしまう場合がある．設定値はデータベースの接続情報などの，固定かつアプリケーション動作に重要な値であるため，一つの設定値の参照間違いが，アプリケーションの誤動作を引き起こす可能性が高くなる．

更に，設定値の参照によって引き起こされるアプリケーションの誤動作は，バグ修正に時間がかかる．なぜならば，アプリケーションが誤動作を起こした場合，設定ファイルとPHP プログラムの両方を見比べ，どこに原因があるかを探しながらデバッグをしなければならないからである．複数のファイルにまたがってデバッグをするうちに，開発者が混乱してしまう場合もある．

Oppenheimer らは [7] 3 つの商用インターネットサービスを調査し，サービスの動作ミスを引き起こす可能性のあるバグの 50% 以上は，設定ファイルのミスであることを発見した．実際に報告されたアプリケーションの設定ミスでは，Yahoo!の Zookeeper ゲームでは，設定ミスによりユーザ画面に表示されてはいけない情報が掲載された [8]．Facebook は設定値のミスにより 2 時間サービスが停止 [9]．スウェーデンの全体ドメインである.se は DNS の設定ミスにより 1 時間サービスが停止 [10]．マイクロソフト社のクラウドサービスである Azure は，設定ミスにより 2 時間半サービスが停止 [11]．MongoDB のバージョン 3.0 では，設定ファイルに脆弱性があることが報告されている [12]．

このように，設定値によるアプリケーションの不具合は，アプリケーション開発者だけでなく，サービス利用者や社会にも影響を及ぼす大きな問題となっている．

1.2 研究目的と概要

本研究は PHP5 アプリケーションを対象とした，設定値の参照ミス検知手法を提案する．また実際に提案手法を実現した，設定値の参照ミス検知ツール『Mis.Config』を開発した．

提案手法は，PHP アプリケーションの設定ファイル内で定義された設定値が，PHP プログラム内で参照される時，定義された型と異なる型で参照されたときの型不一致を検知するものである．手法の流れとしては，設定ファイル内の設定値の型を特定し，設定値の型リストを作成する．型の特定には正規表現を用いる．次に，プログラム内で設定値を参照している部分を見つけ，設定値がどの型を想定して参照されているかを解析する．そして，設定値の型リストとプログラム内の設定値の型を比較し，型が一致していれば正常，型が不一致していればその旨を報告する．

また，提案手法を実現した，設定値の参照ミス検知ツール『Mis.Config』を開発した．手法の実現のため，プログラム解析にはコントロールフローグラフとシンスライシングを用いた．Mis.Config の参照ミス検知能力を検査するため，PHP アプリケーションに埋め込まれた設定値の参照時型不一致を検出できるか精度実験をおこなった．解析対象のアプリケーションは，SourceForge で公開されているオンラインチェスゲーム WebChess0.9.0

を用いた．WebChess の設定値を，情報系学生 4 人に自由に変更・追加をしてもらい，型不一致が起きている場合，Mis.Config がそれらを検知できるかを調査した．

結果，本手法が対象とした整数型・文字列型・論理型に変更された型不一致は全て検出することができた．手法全体の精度は，適合率が 100%，再現率が 78%，F 値が 85% となった．更に Mis.Config を用いて，実際に報告された参照時の型不一致バグを再現できるかを試し，再現できることを確認した．

1.3 本論文の構成

本論文の構成として，2 章で実際に報告されたバグ事例を紹介し，問題意識について論じる．3 章で関連研究を紹介する．4 章では本手法の実現にあたって必要な背景知識を説明する．5 章では本手法の概要と解析の流れを順を追って説明し，6 章では本手法の実装方法について説明する．7 章では手法をもとに開発したツールの精度実験の概要とその結果，考察を論じる．8 章では議論と今後の課題について論じ，9 章で本論文をまとめる．

第 2 章

実アプリの設定値参照ミス関するバグ報告

本章では，公開されている PHP アプリケーションで実際に発生した，設定値の参照時型不一致ミスによるバグ報告を紹介する．

2.1 Joomla! 1.5

Joomla![*1]は 75 万回以上のダウンロード実績を誇る，最も利用されているコンテンツマネージメントシステム（CMS）の一つである．コンテンツマネージメントシステムとは，WEB サイトの構築・編集・管理を容易にするシステムのことである．Joomla!1.5 で，設定値の参照時型不一致ミスによるアプリケーションの誤動作が報告された [13].

バグの報告によると『 $error_reporting という設定値は文字列型として定義されているが，プログラム内では整数型として参照されている．結果，この参照時の型の不一致によって，アプリケーションが誤動作を起こした.』という内容である．

Joomla!1.5 には$error_reporting という設定値があり，開発者がアプリケーションのエラーレポートを表示させるか否かを設定するフラグの役割を担っている．設定ファイル内では，$error_reporting には初期値として '0' または '-1' が格納されている．このとき，$error_reporting の型は数値がシングルクォーテーションで囲まれているため，文字列型として扱われる．

```
var error_reporting = '-1'; // or '0'
```

[*1] https://www.joomla.org

$error_reporting は includes /framework.php で参照されている．下記にソースコードを示す．

```
1 // System configuration
2 CONFIG = new JConfig();
3 if (@CONFIG->error_reporting === 0) {
4   error_reporting(0);
5 } else if (@CONFIG->error_reporting > 0) {
6   error_reporting(CONFIG->error_reporting);
7   ini_set('display_errors', 1);
8 }
```

ソースコードの 3 行目では，$error_reporting は===の演算子を用いて，0 と比較されている．===の比較演算子は，PHP では値と型の両方が一致しているかを比較する演算子である [14]．0 は整数型であり，比較演算子は===であることから，$error_reporting は整数型と想定して参照されていることがわかる．ソースコードの 5 行目でも，$error_reporting は整数型 0 と，大小を比較されているので，同じく整数型と想定して参照されている．

以上から，$error_reporting は文字列型として定義されているが，プログラム内では整数型として参照されており，設定値の参照時に型の不一致が発生していることがわかる．これにより，条件分岐文で分岐が正常におこなわれず，設定したフラグ通りにアプリケーションが動作しないバグが発生した．

2.2 WebChess

WebChess[*2]は，オープンソースソフトウェアのダウンロードサイトである SourceForge[*3]上で，10 年以上公開され続けているオンラインチェスゲームである．WebChess は PHP で開発されており，開発者は SourceForge からプログラムをダウンロードし，自身の WEB サーバに WebChess を設置することで，オンラインのチェスゲームを楽しむことができる．WebChess のバージョン 1.0.0 では，設定値のミスによってアプリケーションの誤動作が発生するバグが報告された [15]．

バグ報告内容は『設定値$CFG_SESSIONTIMEOUT は整数型ではなく文字列型に変更すべきである』と言う旨である．しかし$CFG_SESSIONTIMEOUT が参照されているプログラムを読むと，$CFG_SESSIONTIMEOUT は大小比較で参照されているため，整数型であること

[*2] https://sourceforge.net/projects/webchess/

[*3] https://sourceforge.net

が想定される．（下記コード 11 行目）よって，$CFG_SESSIONTIMEOUT を文字列型に変更する必要はなく，誤ったバグ報告がなされたことになる．

```
1  <?
2    /* load settings */
3    if (!isset($_CONFIG))
4      require 'config.php';
5
6    if (!isset($_SESSION['playerID']))
7      $_SESSION['playerID'] = -1;
8
9    if ($_SESSION['playerID'] != -1)
10   {
11     if (time() - $_SESSION['lastInputTime']  >= $CFG_SESSIONTIMEOUT)
12       _SESSION['playerID'] = -1;
13     else if (!isset($_GET['autoreload']))$_SESSION['lastInputTime'] = time
           ();
14   }
15
16   if ($_SESSION['playerID'] == -1)
17     die("Session timed out. Please<a href='index.php'>login again</a>to
           continue.");
18 ?>
```

実際の原因は，設定値の参照時型不一致が原因ではなく，チェスゲームのインストール機能にあった．WebChess1.0.0 は平易化のため，ゲームのインストール機能が提供されている．図 2.1 は，チェスゲームインストール画面のキャプチャである．ユーザは図 2.1 のインストール画面から，チェスゲームのパラメータを設定し，ゲームをインストールする．

インストール機能を提供する install.php をみると，インストール画面で設定されたパラメータはそのまま，設定ファイルの対応する箇所に入力されるというプログラム内容である．ユーザによってパラメータが指定されなかった場合，設定値には' 'が格納されることとなり，設定値は自動的に文字列型として定義されてしまう．これにより，例えばゲームプレイ人数のパラメータは整数の入力を想定しているにもかかわらず，パラメータが無記入のままだと，自動的に設定値は文字列型になってしまうため，アプリケーションの誤動作が発生する．このように，開発経験のあるプログラマーであっても，設定値をめぐるバグの根本的な原因を発見することは難しい場合がある．

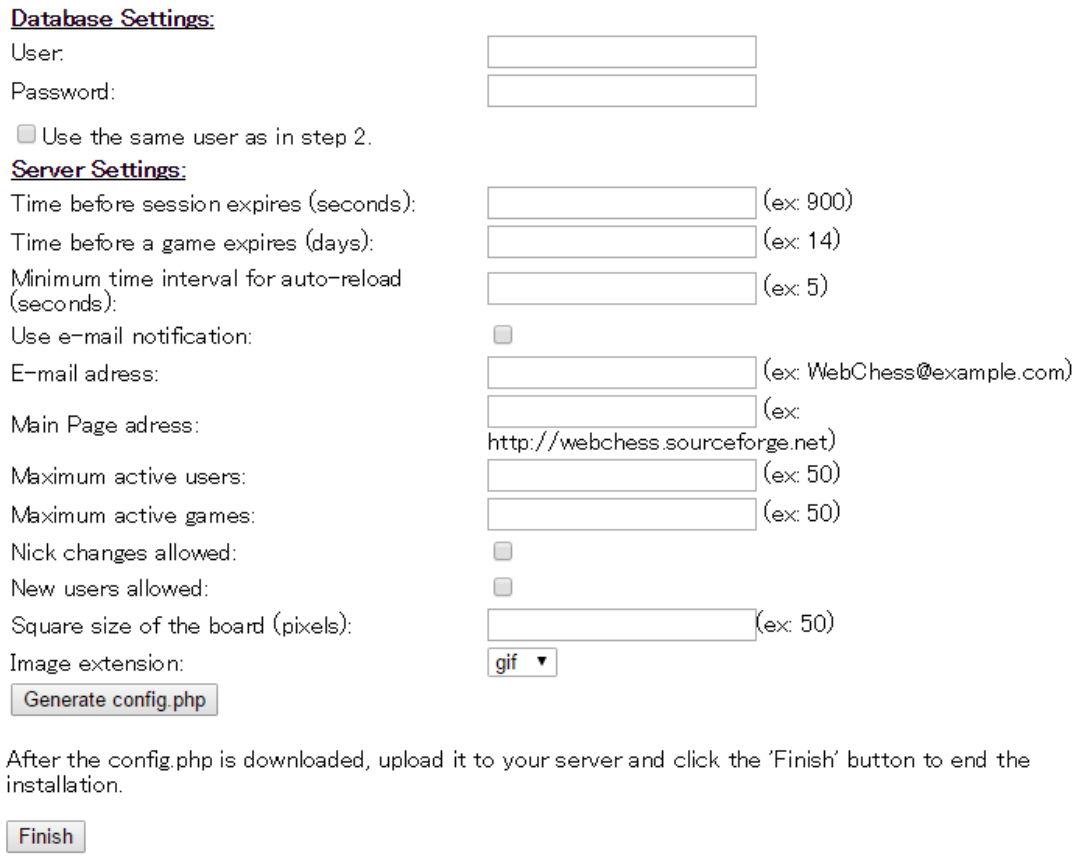

図 2.1　WebChess のチェスゲームインストール画面

2.3　参照ミスが発生しうる設定値の例

Resopnsive Filemanager[*4] は GitHub[*5] で公開されている PHP アプリケーションである．Resopnsive Filemanager の設定ファイルには`$copy_cut_max_size` という設定値がある．`$copy_cut_max_size` には，数値もしくは FALSE を設定するように指示されており，2 つの型になることが想定できる．

```
// defines size limit for paste in MB / operation
// set 'FALSE' for no limit
$copy_cut_max_size    = 100;
```

[*4] http://www.responsivefilemanager.com
[*5] https://github.com

プログラム内では，$copy_cut_max_size は 2 つの型の場合を想定した，条件分岐文になっていることがわかる．（下記コード 5 行目）一つの設定値が複数の型を許す場合，開発者の混乱を引き起こし，設定値の参照時に，定義していないほうの型を想定したプログラムを書いてしまうなど，型不一致のバグが発生しうる可能性がある．Eshkevari らは，PHP アプリケーションの変数の型変更について，ほとんどの型変更は不要であり，ローカル変数やリファクタリングによって置き換えることができることを示している [17]．$copy_cut_max_size も，複数の型を一つの設定値にまとめるのではなく，例えば FALSE は-1 にするなどの対策が考えられる．

```
1 <?php
2 // size over limit
3 if ($copy_cut_max_size !== FALSE && is_int($copy_cut_max_size)){
4   if (($copy_cut_max_size * 1024 * 1024)  < foldersize(path)){
5     die(sprintf(lang_Copy_Cut_Size_Limit,
6       ($_POST['sub_action'] == 'copy' ?  lcfirst(lang_Copy) : lcfirst(
            lang_Cut)),
7       $copy_cut_max_size));
8    }
9 }
10 ?>
```

第 3 章

関連研究

M. Attariyan らは，ソフトウェアの設定ミスの原因をつきとめる手法の提案と，手法をもとにした設定ミスの原因検知ツール『ConfAid』を開発した [23][24]．ConfAid の特徴は，プログラム解析時，バイナリから静的にコントロールフローグラフを生成することである．バイナリからコントロールフローグラフを生成することで，解析対象のプログラミング言語を限定する必要がなく，多くのソフトウェアを解析することができる．ConfAid を用いて，OpenSSH[*1]・Apache[*2]・Postfix[*3]で実際に生じた設定ミスの原因を突き止められるか精度実験をおこない，発見できたことを確認した．しかし，ConfAid は一つのソフトウェアの解析に時間がかかる課題がある．

Zhang らは，Java アプリケーションで発生したバグが，設定値の参照ミスによるものであるかを検知する手法の提案・手法をもとにした ConfSuggester を開発した [20] ．Zhang らは，ソフトウェアバージョンアップ時のソースコード変更によるバグ発生の多さに着目し，バージョンアップ時のソースコードを中心にプログラム解析をおこなっている．ConfSuggester は，ソースコード変更時に発生した設定値参照の変更等から，設定値の変更がバグの原因であるかを検知する．ConfSuggester はプログラム解析の際，Java アプリケーションのバイトコードを解析する．

Nadi らは，C 言語の設定制約を自動的に抽出する手法を提案した [21]．X. Xia らは，バグ報告の自然言語による静的モデル解析によって設定ミスを見つける手法を提案し，手法の精度実験では，5 つのオープンソースソフトウェアで実際に報告された 3,203 個のバグを対象に，設定ミスを見つけることが出来るかを実験した [22].

[*1] https://www.openssh.com/

[*2] https://www.apache.org/

[*3] http://www.postfix.org

設定ファイル値の設定ミスに関する研究は，Java や C 言語については既存研究で提案されている [18] [19] [5].

以上のように，設定値の参照ミスや，プログラムのバグの原因が設定値であるかを検知する手法は提案されてきた．しかし，本研究が対象としている，設定値の型不一致によるバグを検知する手法は，これまで提案されていない．

第 4 章

背景知識

4.1 PHP（Hypertext Preprocessor）

PHP はオープンソースの汎用スクリプト言語である [2]．従来の静的な HTML を拡張し，より高度な WEB 開発を目指すため，1994 年につくられた．PHP の正式名称は Hypertext Preprocessor であり，略語の PHP は正式名称の再帰的頭字語である．2017 年 1 月現在，82.5% のサーバーサイドプログラミング言語を使用している WEB サイトは PHP によって開発されており，非常にシェアの高いプログラミング言語である [3].

PHP はサーバーサイドプログラミング言語であるため，PHP によって開発されたアプリケーションはクライアントサーバモデルにのっとったシステムとなる．クライアントサーバモデルとは，アプリケーションの利用者であるクライアントが，必要な情報をサーバーに問い合わせ，サーバは要求された情報を返したり表示したりすることで成り立つ方式である．

一般的にクライアントサーバモデルは，サーバを制御するサーバサイドのプログラミング言語と，クライアントを制御するプログラミング言語を分けて開発されているため，しばしばこのモデルを利用したソフトウェアの保守性の難しさについて議論されてきた [4].

Sayagh らは，PHP を用いて開発されたアプリケーションで，最も利用されている WordPress の保守性について調査した [5]，WordPress はコンテンツマネージメントシステム（CMS）と呼ばれる，WEB サイトの構築・編集を容易にするシステムである [6]. WordPress はプラグインという，ユーザが拡張したい機能を簡単に追加できる機能がある．このプラグイン機能の利便性が，CMS の中で WordPress が最も利用されている理由の一つである．

プラグインはそれぞれ設定ファイルを持ち，自身のプラグインを制御するためのパレ

メータ情報を保存する．Sayagh らは，ひとつの WordPress が複数のプラグインを所有している時，設定ファイルの競合による保守性が担保されない可能性について調査した．

結果，484 個のプラグインのうち，35 個のプラグインが WordPress のデータベース設定項目を，直接上書きしていた．調査したプラグインの 18% にあたる 89 個のプラグインは，一つもしくは複数のデータベース設定項目を，間接的に上書きしてた．この結果から，設定ミスが発生した場合，原因を特定するために複数の設定ファイルを調査する必要があることがわかる．

Nguyen らは，PHP アプリケーションでは，ファイル間の変数参照が非常に高いことを示した [29]．こちらに関しても，ファイル間をまたがる変数がバグの原因である場合，バグの原因を探すために複数のファイルや情報の流れを検査する必要があり，デバッグに時間がかかることが想定できる．

4.2 スライシング

スライシングはプログラムの解析手法であり，プログラム内の基準変数 C の実行に影響するプログラムをスライス結果として出力するものである．

4.2.1 フルスライシング

1981 年に Weiser によって提唱された [26]．フルスライシングは，基準 C の実行に影響する部分の定義を「基準変数 C の評価に影響を及ぼすあらゆるプログラム文 T はスライシングの結果に現れるべき」とした．

4.2.2 シンスライシング

シンスライシングは，2007 年に Sridharan らによって提唱された [27]．シンスライシングは，基準 C の実行に影響する部分の定義を，「基準変数 C はプログラム文 T のプロデューサー（生産者）である．もし，C が T の値を評価・コピーする処理に必要な連続の一部であれば，スライシングの結果に含む」とした．

4.2.3 フルスライシングとシンスライシングの解析結果の比較

図 4.1 のプログラムを例に，フルスライシングとシンスライシングの解析を比較する．図 4.1 は入力されたフルネームから，名前を表示するプログラムである．しかし，John

```
1  class Vector {
2    Object[] elems; int count;
3    Vector() { elems = new Object[10]; }
4    void add(Object p) {
5      this.elems[count++] = p;
6    }
7    Object get(int ind) {
8      return this.elems[ind];
9    } ...
10 }
11 Vector readNames(InputStream input) {
12   Vector firstNames = new Vector();
13   while (!eof(input)) {
14     String fullName = readFullName(input);
15     int spaceInd = fullName.indexOf(' ');
16     String firstName =
         fullName.substring(0,spaceInd-1);
17     names.add(firstName);
18   }
19   return firstNames;
20 }
21 void printNames(Vector firstNames) {
22   for (int i = 0; i < firstNames.size(); i++) {
23     String firstName = (String)firstNames.get(i);
24     print("FIRST NAME: " + firstName);
25   }
26 }
27 void main(String[] args) {
28   Vector firstNames =
       readNames(new InputStream(args[0]));
29   SessionState s = getState();
30   s.setNames(firstNames);
31   ...;
32   SessionState t = getState();
33   printNames(t.getNames());
34 }
```

図 4.1　シンスライシングとフルスライシングの比較

Doe と入力すると，プログラム 24 行目では John ではなく『Joh』が出力されてしまう．このバグは 16 行目で発生しており，入力されたフルネームから名前を抜き出す際の文字数指定を誤ったためである．

バグの原因を発見するため，24 行目の firstName を基準変数 C として，フルスライシングとシンスライシングを図 4.1 に適用する．フルスライシングの定義にしたがってプログラムを解析すると，出力結果には図 4.1 に表示される全てのコードが出力される．シンスライシングの定義にしたがってプログラムを解析すると，出力結果には図 4.1 の下線部分が出力される．

フルスライシングの結果は，バグの原因をつきとめるための手がかりとしては範囲が広く，原因をつき止めることは難しい．シンスライシングの結果は，より範囲の狭いものとなり，バグの原因となっている 16 行目も含んでいるため，デバッグがし易くなったことがわかる．本研究では，基準変数 C により深刻な影響を及ぼす部分のみを出力することができる，シンスライシングを採用する．

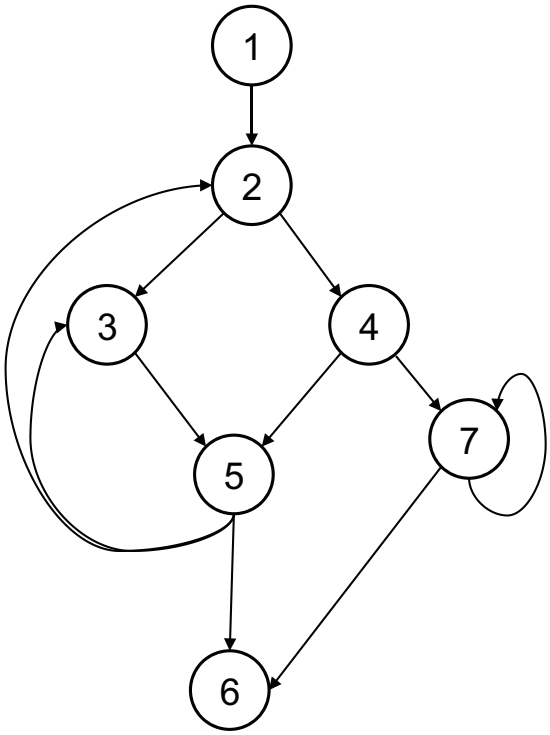

図 4.2　コントロールフローグラフ

4.3 コントロールフローグラフ

コントロールフローグラフは，1970 年に Allen によって提唱されたもので，プログラムが通る可能性のある経路を，網羅的にグラフで表現したものである [25]．プログラムの各遷移をノードで表現し，実行の順番によってノード間を関連付ける．下記の内容をもつコントロールフローグラフを図 4.2 に示す．

1. ノード 1 の次の実行先はノード 2 である
2. ノード 2 の次の実行先はノード 3 またはノード 4 である
3. ノード 3 の次の実行先はノード 5 である
4. ノード 4 の次の実行先はノード 5 またはノード 7 である
5. ノード 5 の次の実行先はノード 2・ノード 3・ノード 6 のいずれかである
6. ノード 6 の次の実行先はない
7. ノード 7 の次の実行先はノード 6 またはノード 7 である

4.4 適合率・再現率・F 値

表 4.1　ツールの評価指標

	バグである	バグでない
手法で検出	TruePositive(TP)	FalsePositive(FP)
手法で未検出	FalseNegative(FN)	TrueNegative(TN)

精度実験の評価指標には，適合率・再現率・F 値を用いる．

適合率は手法の正確性をあらわす．手法が検出した設定値の型の不一致（バグ）が本当にバグである割合をしめす．（式 4.1）

再現率は手法の網羅性を示す．これらの値は，精度実験対象のプログラムに埋め込まれたバグを，本手法が検出できた割合を示す．（式 4.2）

F 値は，正確性と網羅性の総合的な評価を示すものである．F 値は，適合率と再現率の調和平均である．（式 4.3）

適合率・再現率の算出に用いる要素を，表 4.1 に示す．TruePositive(TP) は，本手法が検出したバグが，実際にバグである個数を示す．FalsePositive(FP) は，本手法が検出したバグが，実際はバグではない個数を示す．FalseNegative(FN) は，実際はバグであるの

に，本手法が検出しなかった個数を示す．TrueNegative(TN) は，型が一致している設定値，つまりバグではないものを，手法が検出しなかった個数を示す．

$$\text{適合率} = \frac{TP}{TP + FP} \tag{4.1}$$

$$\text{再現率} = \frac{TP}{TP + FN} \tag{4.2}$$

$$F\ \text{値} = \frac{2 \times \text{適合率} \times \text{再現率}}{\text{適合率} + \text{再現率}} \tag{4.3}$$

第 5 章

提案手法

5.1 提案手法の概要

本研究では，PHP アプリケーションの設定ファイル内で定義された設定値の型と，その設定値がプログラム内で参照されたときの型不一致を検出する手法を提案する．

本手法のゴールは，設定値が参照された際の型不一致を検出することである．型不一致とは，PHP アプリケーションの設定ファイル内で定義された設定値の型が，プログラムで参照される時に定義と異なる型で参照されることである．本手法は PHP5 系で開発されたアプリケーションを対象とする．

提案手法の流れを図 5.1 に示す．まず，設定ファイル内の設定値の型を，独自のルールに則って特定する．そして，特定した設定値とその型をペアとして保存する，設定値型リストを作成する．作成したリストは，後の型不一致検査に利用する．次に，PHP プログラム内で設定値を格納しているローカル変数を特定する．本手法では，設定値を格納しているローカル変数も設定値とみなし，ローカル変数も型不一致の検査対象とする．その後，PHP プログラム内で参照されている設定値の型と，設定値を格納しているローカル変数の型を，始めに作成した設定値型リストと照合する．照合の結果，型が一致しなければ，設定値の参照時に型不一致が発生している旨を報告する．

5.2 設定値の型を特定する

設定値がどの型で定義されているかを特定する．本手法が対象とする PHP5 は，変数の宣言時に型を決める必要がない．PHP は動的型付け言語であり，変数に格納された値を基準にして，動的に型が付けられる．よって，あらかじめ設定値の型を特定し，型一致

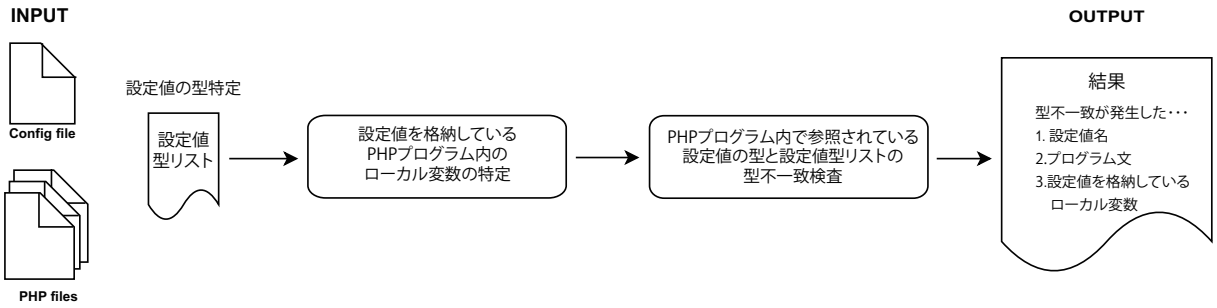

図 5.1　提案手法の流れ

表 5.1　PHP アプリケーション設定ファイルの設定値の型特徴

PHP Apps	GitHub Star	LOC	整数型	論理型	文字列型	その他
Mural	9	950	1	2	3	0
Framadate	32	33,627	1	3	0	0
wifidog	117	65,229	0	7	26	0
Youtube-dl-WebUI	129	8,063	0	1	3	0
Resopnsive File-manager	397	26,114	15	26	25	0
nextcloud	407	159,828	13	30	61	0

検査の際に用いる正解データを用意する必要があるため，本手法では独自のルールに則って，設定値の型を特定する．型の特定には正規表現を用いる．

PHP の変数はスカラー型が 4 種類ある．論理値 (boolean)，整数 (integer)，浮動小数点数 (float, double も同じ)，文字列 (string) である [28]．本手法では，論理型，整数型，文字列型で定義されている設定値を対象とする．

この三種類の型を決定するにあたって，公開されている PHP アプリケーションの設定ファイルを調査し，設定値にはどの型が使われているか調査した．調査の結果，PHP アプリケーションの設定値は論理型，整数型，文字列型の三種類で表現されていることがわかった．調査結果を図 5.1 に示す．

調査は，Github 上に公開されている 6 個のアプリケーションを対象とした．PHP アプリケーションの選定にあたり，プログラム内容の多様性をもたせるため，既存の PHP フレームワークを利用しない，フルスクラッチのアプリケーションを対象した．アプリケー

ション名は，Mural [*1]，Framadate [*2]，wifidog [*3]，Youtube-dl-WebUI [*4] ，Responsive Filemanager [*5], nextcloud [*6]である．Github スター数は 9〜407 個, LOC (Line Of Code) は 950〜159,828 行と，幅広い規模のアプリケーションを選定した．以上の調査結果より，本手法ではこの三種類を検査対象とする．

5.3 設定値を格納しているローカル変数の特定

```
<?
  $example = $CFG_SESSIONTIMEOUT;
  if (time() - $_SESSION['lastInputTime']  >= $example)
    $_SESSION['playerID'] = -1;
  }
  else if (!isset($_GET['autoreload'])){
    $_SESSION['lastInputTime'] = time();
  }
?>
```

上記のプログラム例のように，PHP アプリケーションでは，設定値が PHP プログラム内のローカル変数に格納されている場合がある．例では，ローカル変数$example に $CFG_SESSIONTIMEOUT の値が代入されており，IF 文では整数型として参照されていることがわかる．上記の例のように，設定値を代入しているローカル変数も設定値とみなし，参照時型不一致の検査対象とする．

5.4 設定値の型一致検査

型不一致の前準備として，設定値の型をまとめたリスト，設定値を代入しているローカル変数を特定した．最後に，PHP プログラム内で参照されている設定値の型が，設定ファイル内で定義された型と一致しているかを検査する．

本手法は，PHP ファイルを 1 つずつ解析する．PHP プログラムで，設定値または設定値を代入しているローカル変数を参照している文を探し出し，該当プログラム文において，設定値の型が定義と一致しているかを検査する．型一致の検査には，下記の正規表現

[*1] https://github.com/laravolt/mural
[*2] https://framadate.org
[*3] http://dev.wifidog.org
[*4] https://github.com/p1rox/Youtube-dl-WebUI
[*5] http://www.responsivefilemanager.com
[*6] https://github.com/nextcloud/server

リストを用いる．

Integer → .+[−+∗/]CFG_NAME[−+∗/].+

Integer → if(CFG_NAME(>=.+|={2}d+))

Boolean → if(!?CFG_NAME)

Boolean → if(CFG_NAME(={2}|!=){true|false}

String → mysql_connect(CFG_NAME+,? CFG_NAME+,? CFG_NAME+)

String → .+CFG_NAME.+

照合に用いる正規表現は，IF 文を中心にリストを作成した．IF 文に着目した理由は，第 2 章で取り扱った Joomla!と WebChess で報告されたバグが，IF 文で発生していたためである．このことから，本研究が対象とするバグは IF 文で発生しやすいと考えた．また，IF 文で設定値の型不一致が発生した場合，型の不一致により誤った条件分岐がなされる可能性がある．誤った条件分岐による，アプリケーションが誤動作を引き起こす影響は大きい．よって，本研究では IF 文に着目した正規表現リストを作成した．

正規表現リストの解説をする．入力された PHP プログラム文において，設定値が四則演算子で囲まれている場合，設定値は整数型として参照されているとみなす．もしくは，設定値が IF 文内で数量の比較演算子，または数値との一致がなされている場合も，設定値は整数型として参照されているとみなす．

論理型は，IF 文内で設定値が単体で参照されている場合，または論理否定演算子と共に参照されている場合．そして，true や false と比較されている場合である．

文字列型の設定値は，データベースの mysql の接続設定に用いられることが多いため，mysql_connect 関数を対象とした．mysql_connect 関数は，データベース名・ユーザー名・パスワードの入力が必要となる．通常，データベースの接続情報は，アプリケーションの設定ファイルに入力する．アプリケーションにおいて，アプリケーションの情報を管理するデータベースへの正常な接続は重要であるため，データベースの接続情報が文字列型で入力されているか検査することは重要である．

これらの正規表現を該当プログラム文に照合し，一致した正規表現に対応する型を特定する．次に，特定したプログラム内設定値の型と，事前に用意した設定値の型リストを設定値名で照合する．2 つの型を比較して同じ型であれば，設定値の参照時に型不一致は発生しておらず，設定値はプログラム内で正しく参照されていると判断する．対して，2 つの型が一致しなければ，設定値の参照時に型不一致が発生していると判断する．

5.5 解析結果の表示

型一致検査の結果，設定ファイルとプログラム内で設定値の型が一致しなかった場合，本手法は型不一致を報告する．報告内容は，型不一致が発生した設定値名・型不一致が発生したプログラム文・該当設定値を格納しているローカル変数の 3 つである．

第 6 章

Mis.Config：提案手法の実装

本章では，提案手法をもとに開発した，型不一致検出ツール Mis.Config について説明する．

6.1 Mis.Config 概要

Mis.Config は 5 章で提案した PHP アプリケーション設定値の参照時型不一致検査を，ツール化したものである．入力として，解析をした PHP アプリケーションのディレクトリパス，設定ファイルのパスを渡す．解析の結果，型不一致が発生していると判断した場合，1. 設定値の型不一致が発生したプログラム文，2. 設定値の名，3. プログラム内で該当設定値を保持しているローカル変数の 3 つをテキストファイルの形式で返す．

6.2 開発環境

Mis.Config は Eclipse4.5 上で Java8 を用いて開発したプログラム解析には，シンスライシングとコントロールフローグラフを用いる．シンスライシングとコントロールフローグラフの実装にあたって，Nguyen らが提案した PHP アプリケーション解析ツールの VarAnalysis の値を利用した [29][30][31][32] ．VarAnalysis は Java で開発されている．PHP プログラムだけでなく，PHP・HTML・JavaScript 間の値の受け渡しフローの解析を行う．本手法は PHP プログラムのみを解析したいため，VarAnalysis の PHP プログラム解析部分を利用した．

VarAnalysis はプログラム解析の際に，コントロールフローグラフを生成するが，PHP プログラムに case 文・function 文が含まれている場合，そのスコープのコントロールフ

ローグラフを生成しない制限があった．本手法は VarAnalysis を改良し，PHP プログラムの case 文と function 文の部分をプログラム全体から切り出し，スコープ単位でコントロールフローグラフを生成し，プログラム全体のコントロールフローグラフに統合することによって，case 文・function 文部分もコントロールフローグラフが生成出来るようにした．

6.3 コントロールフローグラフの実装

解析の便宜上，VarAnalysis が生成したコントロールフローグラフから，本手法で必要な情報のみを含んだコントロールフローグラフにするため，独自でコントロールフローグラフを実装した．

付録プログラム A.2 の getGraph 関数を呼び出すと，VarAnalysis が生成した PHP プログラムのコントロールフローグラフのうち，本手法に必要な情報を含んだコントロールフローグラフを生成する．本手法が必要な情報は，一行分のプログラムと PHP のファイル名と次の実行先である．VarAnalysis のコントロールフローグラフから必要な情報を取り出し，独自の Node クラスを用いて Node インスタンスを生成する．本手法のコントロールフローグラフのノードが持つ情報を図 6.1 に示す．

▲ [2]	Node (id=127)
index	1922930974
label	2
line	null
offset	null
path	"sysinfo_config.php" (id=283)
statement	"$config_output[]=$cf[$k];" (id=284)
type	"PhpVariableRef" (id=285)
value	"config_output" (id=286)
visited	false

図 6.1　コントロールフローグラフのノード情報

6.4 設定値の型を特定する

正規表現を用いた独自の規則に従って，設定値の型を特定しリストを作成する．作成したリストは，型一致検査の際に，プログラム内で参照されている設定値の型との照合に使用される．本節に対応するコードは付録プログラム A.3 の getOptionsTypeList 関数である．getOptionsTypeList 関数にアプリケーションの設定ファイルパスを入力すると，指定設定ファイル内の設定値とその型を解析し，設定値名とその型のペアをリストにして返す．

6.4.1 設定値の型特定に用いる正規表現リスト

本手法では，正規表現を用いて設定値を解析し，型の特定を行う．表 6.1 に示す正規表現によって特定する．

表 6.1 設定ファイル内の設定値の型特定に用いる正規表現

文字列型	整数型	論理型
”.*” |’.*’	\d+	(true |false)

1. 文字列型：値がシングルまたはダブルクォーテーションマークで囲まれている
2. 整数型：値がクォーテーションマークで囲まれておらず，整数である
3. 論理型：値がクォーテーションマークで囲まれておらず，true または false である

例えば，WebChess0.9.0 の設定ファイルでは，$CFG_USEEMAILNOTIFICATION には false が代入されている．

```
@config.php
CFG_USEEMAILNOTIFICATION = false;
CFG_MAILADDRESS = "WebChess@webchess.org";
CFG_MAXUSERS = 50;
```

本手法で設定値の型特定を行うと，この設定値は「値がクォーテーションマークで囲まれておらず，true または false と書かれている」ため，$CFG_USEEMAILNOTIFICATION は boolean 型として認識される．

同様に，$CFG_MAILADDRESS は「値がシングルまたはダブルクォーテーションマークで囲まれている」ため文字列型である．$CFG_MAXUSERS は「値がクォーテーショ

ンマークで囲まれておらず，整数である」ため整数型である．図 6.2 に設定値型特定の解析画面を示す．

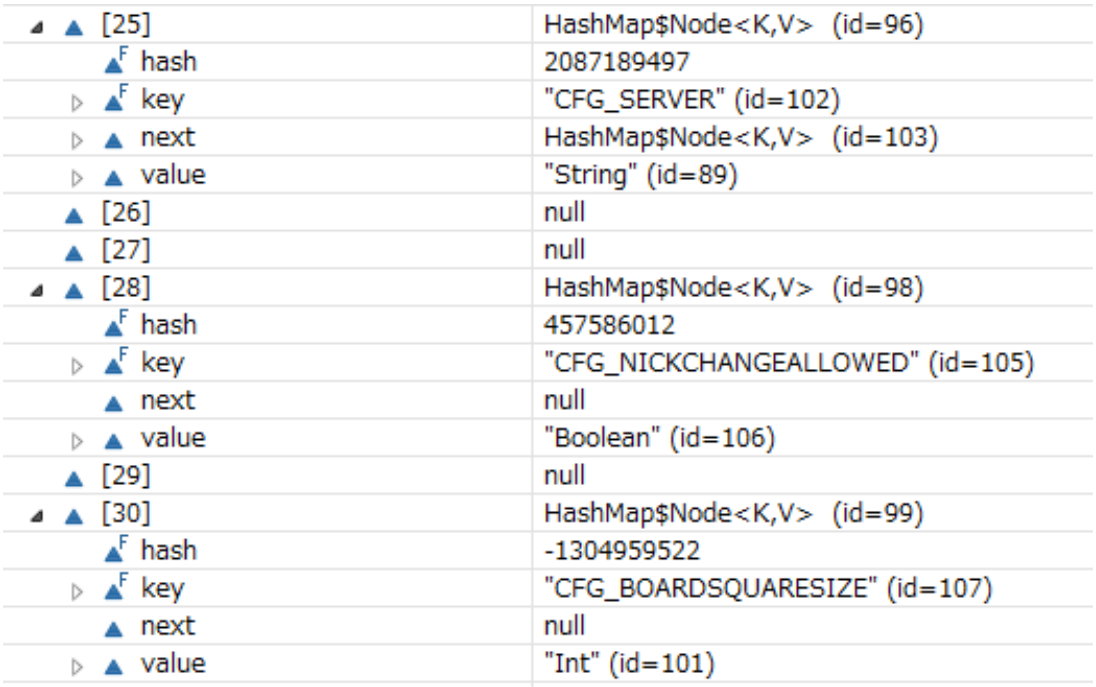

図 6.2　設定値型特定の解析画面

6.5　設定値を格納しているローカル変数の特定

PHP プログラム内の，設定値を格納しているローカル変数を全て特定する．設定値の参照時に型不一致が発生した場合，設定値を格納しているローカル変数の型も，定義した型と一致しない可能性がある．それにより，該当ローカル変数の参照先もアプリケーションが予期しない動きをしている可能性があるため，PHP プログラム内のローカル変数を抽出する．

付録プログラム A.4 の getAffectedValuesList 関数で，設定値を格納するローカル変数を特定する．getAffectedValuesList には，入力として PHP ファイルのスライシング結果と設定値名を渡す．結果として，入力した設定名の値を格納しているローカル変数をリストで返す．

入力で渡す PHP ファイルのスライシング結果は，VarAnalysis の ReferenceDetector クラスに PHP のファイルパスを渡すことで取得することができる．これによって，PHP

プログラムの変数をそれぞれ基準にして，全てにシンスライシングを適用した結果が得られる．適用した結果は，referenceManager.getValuesAndStatementsList を経由して取得する．結果は文字列型のハッシュマップ形式である．

入力のスライシング結果はハッシュマップで，基準変数 C がキーで，シンスライシング結果が要素となっている．シンスライシング結果は，&&で繋がった文字列である．getAffectedValuesList 関数は，シンスライシング結果の文字列を解析し，設定値を代入している変数を見つけ，結果リストに含む．

6.6 設定値の型一致検査

最後に，PHP プログラム内で参照されている設定値の型が，設定ファイル内で定義された型と一致しているか検査する．付録プログラム A.5 の isWellConfigured 関数は，設定値の参照元と先で型が一致しているかを検査する．入力として，型一致が一致しているか調べたい設定値名・設定値の型・PHP プログラム文を渡す．出力として，入力した設定値の型が参照元と先で一致しているかを true または false（論理型）で返す．

入力された PHP のプログラム文を，下記の正規表現リストと照合する．照合の結果，一致した正規表現に対応する型と，入力された設定値の型が一致していれば true，そうでなければ false を返す．

Integer → .+[−+∗/]CFG_NAME[−+∗/].+

Integer → if(CFG_NAME(>=.+|={2}d+))

Boolean → if(!?CFG_NAME)

Boolean → if(CFG_NAME(={2}|!=){true|false}

String → mysql_connect(CFG_NAME+,? CFG_NAME+,? CFG_NAME+)

String → .+CFG_NAME.+

照合に用いる正規表現は，IF 文を中心にリストを作成した．IF 文に着目した理由は，第 2 章で取り扱った Joomla!と WebChess で報告されたバグが，IF 文で発生していたためである．このことから，本研究が対象とするバグは IF 文で発生しやすいと考えた．また，IF 文で設定値の型不一致が発生した場合，型の不一致により誤った条件分岐がなされる可能性がある．誤った条件分岐による，アプリケーションが誤動作を引き起こす影響は大きい．よって，本研究では IF 文に着目した正規表現リストを作成した．

正規表現リストの解説をする．入力された PHP プログラム文において，設定値が四則

演算子で囲まれている場合，設定値は整数型型として参照されているとみなす．もしくは，設定値が IF 文内で数量の比較演算子，または数値との一致がなされている場合も，設定値は整数型型として参照されているとみなす．

論理型は，IF 文内で設定値が単体で参照されている場合，または論理否定演算子と共に参照されている場合．そして，true や false と比較されている場合である．

文字列型の設定値は，データベースの mysql の接続設定に用いられることが多いため，mysql_connect 関数を対象とした．mysql_connect 関数は，データベース名・ユーザー名・パスワードの入力が必要となる [33]．通常，データベースの接続情報は，アプリケーションの設定ファイルに入力する．アプリケーションにおいて，アプリケーションの情報を管理するデータベースへの正常な接続は重要であるため，データベースの接続情報が文字列型で入力されているか検査することは重要である．

6.7 解析結果の表示

```
1  -- chess.php
2  CFG_MINAUTORELOAD
3  if($_SESSION['pref_autoreload']>=$CFG_MINAUTORELOAD)
4  Boolean, false
5
6  -- mainmenu.php
7  CFG_USEEMAILNOTIFICATION
8  <?if($CFG_USEEMAILNOTIFICATION){?>
9  Int, false
```

図 6.3　本手法の結果表示画面

型一致検査の結果，設定ファイルとプログラム内で設定値の型が一致しなかった場合，本手法はエラー部分をテキストファイル形式で，結果を出力する（図 6.3）．出力内容は，型不一致が発生した設定値名・型不一致が発生したプログラム文・該当設定値を格納しているローカル変数の 3 つである．

第 7 章

精度評価実験

本節では，提案手法の精度実験の内容とその結果について説明する．

7.1 精度実験概要

本手法の精度を評価するため精度実験を行った．PHP アプリケーションの設定値を人為的に変更し，PHP プログラム内と設定ファイル間で，設定値の型不一致バグを意図的に発生させる．あるいは設定ファイルに新規の設定値を追加し，PHP プログラム内に埋め込む．そして本手法が，意図的に発生させた型不一致を検出することができるかテストをする．

設定値の変更と追加・プログラムの改変は，情報工学を専攻する学部学生 4 人が実施した．実験の公平性の観点から，本研究に携わっていない学生に依頼した．全員のプログラミング能力は，PHP を使ったメール送信機能や掲示板つきの WEB サイトを制作することができる程度である．学生 4 人には，本手法が「設定値がプログラム内で正しく参照されているかを検知するツール」と伝えており，詳細なアルゴリズムは知らない．

実験対象の PHP アプリケーションは WebChess0.9.0[*1]を採用した．学生 4 人は，WebChess0.9.0 の設定ファイル値を好きなように変更・追加をする．例えば，`$CFG_NEW_USERS_ALLOWED` は，本来 true または false が格納されているため，論理型である．PHP プログラム内でも，`$CFG_NEW_USERS_ALLOWED` が論理型として参照されていることがわかる．

```
@index.php
  if(CFG_NEW_USERS_ALLOWED==true)
```

[*1] https://sourceforge.net/projects/webchess/files/webchess/0.9.0/

本実験で，学生 A が $CFG_NEW_USERS_ALLOWED の値を 123 と設定したと仮定する．この場合，プログラムで参照された設定値の型と，実際の設定値の型が一致しないため，本手法は型不一致を報告する．

7.1.1 実験で変更・追加した設定値

本実験で，学生 4 人が変更・追加した設定値を，表 7.1，7.2，7.3，7.4 にまとめる．

表 7.1 整数型に変更された設定値

設定値名	定義の型	変更後（整数型）
CFG_MINAUTORELOAD	5 (integer)	10
CFG_MAXUSER	50 (integer)	3
CFG_NEW_USERS_ALLOWED	true (boolean)	999 , 3
CFG_EXPIREGAME	14 (integer)	14000
CFG_USEMAILNOTIFICATION	false (boolean)	0 ,100
CFG_MAILADDRESS	'WebChess@webchess.org' (string)	123
CFG_NICKCHANGEALLOWED	false (boolean)	0
CFG_BOARDSQUARESIZE	50 (integer)	50
CFG_BOARDER	-	1
CFG_BBB	-	10000000000
CFG_MINAUTORELOA	-	5

表 7.2 論理型に変更された設定値

設定値名	定義の型	変更後（論理型）
CFG_MINAUTORELOAD	5 (integer)	false
CFG_NICKCHANGEALLOWED	false (boolean)	false
CFG_BOARDSQUARESIZE	50 (integer)	true
CFG_AAA	-	true
CFG_NEW1	-	false

表 7.3　文字列型に変更された設定値

設定値名	定義の型	変更後（文字列型）
CFG_MAXUSER	50 (integer)	3
CFG_NEW_USERS_ALLOWED	true (boolean)	999 , 3
CFG_EXPIREGAME	14 (integer)	14000
CFG_USEMAILNOTIFICATION	false (boolean)	0 ,100
CFG_MAILADDRESS	'WebChess@webchess.org' (string)	123
CFG_NICKCHANGEALLOWED	false (boolean)	0
CFG_BOARDSQUARESIZE	50 (integer)	50
CFG_BOARDER	-	1
CFG_BBB	-	10000000000
CFG_MINAUTORELOA	-	5

表 7.4　その他の型に変更された設定値

設定値名	定義の型	変更後（その他）
CFG_SESSIONTIMEOUT	900 (integer)	900.0 (float)
CFG_MAXACTIVEGAMES	50 (integer)	10.0 (float), echo 'bye'
CFG_NEW_USERS_ALLOWED	true (boolean)	null
CFG_EXPIREGAME	14 (integer)	10.5 (float)
CFG_USEMAILNOTIFICATION	false (boolean)	function foo()print 'hye'
CFG_PASSWORD	'WebChessPassword'(string)	(1,1)
CFG_DATABASE	'WebChess_DB'(string)	$CFG_SERVER
CFG_NEW2	-	1.0 (float)

表 7.5　精度実験

変更後の型	適合率	再現率	F 値
Integer 型	100%	100%	100%
Boolean 型	100%	100%	100%
String 型	100%	100%	100%
その他	100%	6%	11%

7.2 精度実験の結果と考察

実験結果を表 7.5 に示す．

7.2.1 整数型に変更された設定値

$CFG_NEW_USER_ALLOWED は，論理型から整数型に変更されている．index.php で参照されており，条件分岐文で true と比較されているため，設定値は論理型であることが想定されていることがわかる．本手法の型不一致検査により，$CFG_NEW_USER_ALLOWED は参照時に型不一致が発生していることを検知した．

```
@index.php
  if(CFG_NEW_USERS_ALLOWED==true)
```

$CFG_USEEMAILNOTIFICATION は，論理型から整数型に変更されている．

$CFG_USEEMAILNOTIFICATION は，WebChess 0.9.0 で最もプログラム内で参照されている設定値である．プログラム内では論理型であることが想定されている．論理型の変数が条件分岐で利用される場合，下記のコードにように，どの値とも比較しない，もしくは否定の’！’をつける．しかし，設定ファイルでは整数型として定義されているため，本手法で型不一致が発生していることを検知する．

```
@chessdb.php
  if (CFG_USEEMAILNOTIFICATION)
  if (CFG_USEEMAILNOTIFICATION && !isPromoting)
```

$CFG_BBB は，学生によって追加された設定値である．設定ファイルでは，整数型として定義されている．コード内でも, $CFG_BBB は整数型として参照されているため，型不一致は発生していない．

```
@gui.php
  if (board[i][j] == CFG_BBB)
```

$CFG_NEW_USERS_ALLOWED は，論理型から整数型に変更されている．コード内では論理型と想定して参照されているため，本手法は型不一致が発生しているとみなす．

```
@index.php
  if(CFG_NEW_USERS_ALLOWED==true)
```

$CFG_EXPIREGAME は整数型として定義されている．この設定値は，チェスゲームの試合を破棄する日数を指定する．コード内では 2 箇所で参照されている．一つ目は，計算式

で用いられている．二つ目は，変数の値を出力する echo 関数の中で参照されている．本手法は，1 つ目の計算式で用いられている部分のみを，型不一致検査の対象とする．なぜならば，echo 関数は値を出力する関数なので，変数の型に指定はなく，複数の型を許す関数であるため，参照時の型を考慮する必要がないからである．

```
@mainmenu.php
  targetDate = date("Y-m-d", mktime(0,0,0, date('m'), date('d') -
      CFG_EXPIREGAME, date('Y')));
  Games will expire WITHOUT NOTICE if a move isn't made after <? echo (
      CFG_EXPIREGAME); ?> days!
```

$NEW2 は学生によって追加された設定値で，設定ファイルでは整数型として定義されている．値が参照されている chess.php では，整数の 400 と比較がなされており，設定値は整数型として参照されているため，本手法は型不一致検出しなかった．

```
@chess.php
  if(tmpIsValid && 400 == (100*NEW2))
```

$CFG_MAILADDRESS は文字列型から整数型に変更されている．しかし，本手法はこの変更による型不一致を検出しなかった．コード内でこの設定値は文字列の中で参照されており，文字列の中ではどの型であってもただしく出力することができるからである．

```
@chessutils.php
  headers.="From:WebChess<".CFG_MAILADDRESS.">";
  headers.="Reply-To:WebChess<".CFG_MAILADDRESS.">";
```

$CFG_MAXACTIVEGAMES は，浮動小数点数型から整数型に変更されている．しかし，この設定値はコード内では使用されていなかったため，本手法は型不一致検査を実施しなかった．

7.2.2 論理型に変更された設定値

$CFG_MINAUTORELOAD は，整数型から論理型に変更された．この値はチェスゲームの自動読み込み秒数を設定する．初期値は 5 である．コード内では，$CFG_MINAUTORELOAD は下記のコードのように，大小比較に用いられており，整数型であることが想定される．本手法の型不一致検査によって，この値は参照時に型不一致が発生していることが検知される．

```
@chess.php
  if ($_SESSION['pref_autoreload'] >= $CFG_MINAUTORELOAD)
```

$NEW は学生によって追加された設定値で，設定ファイル内では論理型として定義されている．プログラム chess.php では，TRUE と比較されているため，この値は論理型であることが想定される．本手法では，定義された型と参照時に想定されている型が一致しているため，型不一致検査では検出しなかった．

```
@chess.php
  if(isUndoing && NEW != TRUE)
```

$CFG_AAA も，学生によって追加された設定値であり，論理型として定義されている．この値は mainmenu.php で参照されている．コード内では，$CFG_AAA は計算式の中で参照されていおり，整数型であることが想定され，定義と参照時の型が不一致していることがわかる．

```
@mainmenu.php
  targetDate = date("Y-m-d", mktime(0,0,0, date('m'), date('d') - CFG_AAA,
       date('Y')));
```

$CFG_BOARDSQUARESIZE は整数型から論理型に変更されている．本手法は，この変更は型不一致検査をする必要がないとし，型不一致を検出しなかった．コード内でこの設定値は，HTML の出力のため，値を出力する echo 関数の中で参照されている．echo 関数は，どの型の変数でも参照されうる関数である．よって，本手法は参照時の型不一致による影響はないとして，検出をしない．

```
@gui.php
  echo(tmpALT.".gif' height='CFG_BOARDSQUARESIZE' width='
      CFG_BOARDSQUARESIZE' border='0' alt='".tmpALT."'>");
```

7.2.3 文字列型に変更された設定値

$CFG_SESSIONTIMEOUT は整数型から文字列型に変更された．この値は，チェスゲームのプレイタイムアウト時間を秒数で設定する．デフォルトは 900 である．コード内で，この値は大小比較に用いられているので，整数型として参照されていうことがわかる．この変更によって参照時に型不一致が発生しているため，本手法は型不一致を検出した．

```
@sessioncheck.php
  if (time() - _SESSION['lastInputTime'] >= CFG_SESSIONTIMEOUT)
```

$CFG_MINAUTORELOA は学生によって追加された設定値で，設定ファイル内では文字列型として定義されている．この値は index.php で参照されており，数値 5 と比較されてい

るので，整数値として参照されていることがわかる．よって，参照時に型不一致が発生しているため，本手法はこの型不一致を検出した．

```
@index.php
<?php
if(CFG_MINAUTORELOA == 5){
  <title>WebChess Login</title>
}
?>
```

$CFG_CFG は学生によって追加された設定値で 'cfgcfg' の値が代入されているため，文字列型である．コード内では index.php で参照されており，値を出力する echo 関数内で呼び出されている．echo 関数はあらゆる型の変数をとりうるため，本手法は型不一致による影響は少ないと判断し，型不一致を検出しない．

```
@index.php
<?php echo CFG_CFG;
```

$CFG_MAXUSER は，整数型から文字列型に変更された．しかし，コード内では参照されていなかった．$CFG_MAXUSER は，整数型から文字列型に変更された．しかし，コード内では参照されていなかった．$CFG_MAXACTIVEGAMES は，整数型から文字列型に変更された．しかし，コード内では参照されていなかった．

7.2.4 その他の値に変更された設定値

$CFG_SESSIONTIMEOUT は，整数値型から浮動小数点数型に変更されている．コード内では，この設定値は大小比較はの際に用いられているので，数値型であることが想定されている．よって，型不一致が発生している．しかし，本手法は型不一致検査でこの不一致を検出することができなかった．理由としては，本手法は浮動小数点数型を対象としておらず，プログラム内の設定値の型を特定することができず，検査時に見逃してしまった．

```
@sessioncheck.php
  if (time() - _SESSION['lastInputTime'] >= CFG_SESSIONTIMEOUT)
```

$CFG_NEW_USERS_ALLOWED は，設定ファイルで 'null' が格納されていた．コード内では，true と比較されているため，論理型であることが想定されるため，型不一致が発生している．しかし，本手法は型不一致を検出することができなかった．これは，設定ファイル内の設定値を解析する際，値が null だった場合の型を決めていなかったため，解析することができなかった．よって，型不一致検査の照合で，設定値がリストに含まれておらず，検査することができなかった．

```
@index.php
  if(CFG_NEW_USERS_ALLOWED==true)
```

$CFG_DATABASE は $CFG_SERVER の値が格納されている．プログラムでは connectdb.php で参照されている．mysql_select_db 関数は，mysql データベース名を指定するもので，文字列型の変数を入力する必要がある [34]． $CFG_DATABASE には $CFG_SERVER が代入されているため，文字列型であるため型不一致は発生しない．しかし，本手法は変数が変数の値を格納している場合の，解析をすることを想定していない．よって，設定ファイルの設定値解析の際，$CFG_DATABASE の型を特定することができなかった．

```
@connectdb.php
  mysql_select_db (CFG_DATABASE);
```

$CFG_MAXACTIVEGAMES は，整数型から関数に変更されている．

$CFG_USEMAILNOTIFICATION もまた，論理型から関数に変更されている．本手法は，設定値に関数が代入されている場合を想定しなかったため，型を特定することができなかった．

$CFG_PASSWORD は，文字列型から型特定ができない値に変更されている．この値は connectdb.php で参照されている．$CFG_PASSWORD は文字列型であることが想定されている．mysql_connect 関数は，全ての引数を文字列型であることを想定しているからである．本手法は，設定ファイルの設定値解析時， (1,1) の型を特定できなかったため，型不一致を検出することができなかった．

```
@connectdb.php
  dbh=mysql_connect (CFG_SERVER, CFG_USER, CFG_PASSWORD)
```

7.2.5 Joomla!1.5 で報告された型不一致ミスの再現

本手法を用いて，2.1 で解説した Joomla!1.5 で報告された，設定値参照時の型不一致ミスを再現した．

報告されたバグは， $error_reporting が文字列型として定義されているが，プログラム内では整数型であることを想定して参照されており，アプリケーションが開発者の意図しない動きをしたという内容である．

本手法を用いて報告されたバグを再現できるか実験をした．対象とした Joomla!のバージョンは 1.5.26 である． $error_reporting は 3 つの PHP ファイルで参照されている．

3 つのファイルにおいて，$error_reporting は同じ文章で参照されている．

対象設定値が参照されている PHP は以下のとおりである

1. administrator/includes/framework.php
2. includes/framework.php
3. xmlrpc/includes/framework.php

$error_reporting は，設定ファイルでは文字列型で 'true' または 'false' が代入されている．

```
var error_reporting = '-1'; // or '0'
```

プログラム内では，数値 0 との比較がなされているため，整数値であることを想定して参照されている．よって，設定値の参照時に型不一致が発生している．

```
1 // System configuration
2 CONFIG = new JConfig();
3 if (@CONFIG->error_reporting === 0) {
4   error_reporting(0);
5 } else if (@CONFIG->error_reporting > 0) {
6   error_reporting(CONFIG->error_reporting);
7   ini_set('display_errors', 1);
8 }
```

図 7.1 に, 本手法を Joomla!1.5.26 に適用した結果を示す．本手法は, $error_reporting が型不一致を発生していることを検出することができた．

```
1  -- includes/framework.php
2  error_reporting
3  if(@$CONFIG->error_reporting===0){
4  String, false
5
6  -- includes/framework.php
7  error_reporting
8  }elseif(@$CONFIG->error_reporting>0){
9  String, false
10
11 -- xmlrpc/includes/framework.php
12 error_reporting
13 if(@$CONFIG->error_reporting===0){
14 String, false
15
16 -- xmlrpc/includes/framework.php
17 error_reporting
18 }elseif(@$CONFIG->error_reporting>0){
19 String, false
20
21 -- administrator/includes/framework.php
22 error_reporting
23 if(@$CONFIG->error_reporting===0){
24 String, false
25
26 -- administrator/includes/framework.php
27 error_reporting
28 }elseif(@$CONFIG->error_reporting>0){
29 String, false
```

図 7.1　Joomla! 1.5 で報告されたバグの再現画面

第 8 章

議論と今後の課題

本研究では，PHP アプリケーションには，設定値がプログラム内で参照された時の型と，設定ファイル内で定義された型の不一致により，アプリケーションに誤動作が発生するバグがあることに着目した．そして対象の不具合を検知する，参照時の型不一致ミスを検知する手法の提案と，手法を実現した Mis.Config を開発した．

本手法では，文字列型・論理型・整数型の 3 つの型を，型不一致ミス検査の対象とした．

本手法では，設定ファイル内・PHP プログラム内の設定値の型特定には，正規表現を用いた．設定ファイル内の設定値型特定には正規表現で十分であった．PHP プログラム内の設定値型特定は，IF 文を中心とした正規表現リストを作成し，照合に利用した．本研究が対象としているバグは，我々が観測した限りでは IF 文で発生しているため，IF を中心とした正規表現リストを作成した．しかし，例えば，設定ファイルに記入してあったメールアドレスをメール送信関数で利用した場合，メール送信関数で型不一致が発生しても，本手法は検出することができない．よって，PHP プログラム内の設定値型特定に関しては，正規表現リストを増やす，もしくは他の手法を利用することで，より多くの型不一致を検出することができる．

精度実験では，対象としている三種の型間での型不一致は，全て検出することができた．一方で対象としなかった，浮動小数点数型・関数・他変数の代入・null の再現率は低かった．第 5 章 2 節で説明した PHP アプリケーション設定値の型調査結果から，設定値に浮動小数点数型の値・関数が代入される可能性は限りなく低い．しかし，null や他変数の代入は，設定ファイル内で発生する可能性が高い．今後は，null・他変数の代入があった場合も，どのような値が入力され，どのように分類するかを定め，設定値型リストの精度向上につとめたい．

また本手法は，PHP プログラム内で設定値を格納しているローカル変数があった場合，

そのローカル変数も設定値としてみなし，型不一致検査の対象とする機能を備えている．しかし，精度実験に用いた WebChess0.9.0 は，PHP プログラム内で設定値を格納するローカル変数がなかったため，本機能の有効性を示すことができなかった．今後は，そのようなプログラムを含むアプリケーションを対象とした実験を行いたい．

第 9 章

まとめ

これまでソフトウェアの設定ミスに起因するバグ検知のため，様々な手法が提案されてきた．PHP5 で開発されたアプリケーションは，設定ファイル内の設定値がプログラム内で参照されるとき，設定ファイル内で定義された型とは異なる型を想定して参照されることがあり，これによってアプリケーションが誤動作を引き起こすバグが報告されている．

本研究は，PHP5 で開発されたアプリケーションの設定値に関する不具合に着目した．そして対象の不具合を検知する手法の提案と，検知ツールの Mis.Config を開発した．本研究では対象とするバグを「設定値参照時の型不一致」と呼ぶ．

提案手法では，設定値参照時の型不一致を検知するため，設定ファイル内の設定値の型と，PHP プログラム内の設定値の型を照合し，不一致だった場合にその旨を報告する．PHP5 は変数の型を推論しないため，本手法は独自のルールに則って設定値の型を特定する．また本手法では，PHP プログラム内で設定値を格納しているローカル変数も設定値とみなし，型不一致検査の対象とした．

型不一致検査の対象とする型は，文字列型・整数型・論理型の三種類である．対象とする型を決めるため，GitHub 上に公開されている PHP アプリケーション 6 つを対象とし，設定値の型の特徴調査をおこない，全てのアプリケーションが三種類で表現されていたためである．

更に本研究では，提案手法をもとにした，型不一致検知ツール『Mis.Config』を開発した．Mis.Config は，型不一致を検知したい PHP アプリケーションを入力すると，結果として型不一致の有無，型不一致が発生した設定値名と該当プログラム文を報告する．PHP プログラムの解析にはコントロールフローグラフとシンスライシングを用いることで，ツールを実現した．

Mis.Config の精度実験として，実際に公開されている PHP アプリケーションの設定値

を変更・追加し，人為的にプログラム内と設定ファイル間で，設定値の型不一致を発生させた．そして Mis.Config が，これらの型不一致を検知できるか精度実験を行った．結果，対象とした三種類の型間での型不一致は，再現率・適合率共に 100% であり，全て検出することができた．一方で，想定しなかった値や，対象としなかった浮動小数点数型の型不一致は検出することができなかった．全体の精度は，適合率 100%, 再現率 78%, F 値 85% であった．

本研究は，PHP5 で開発されたアプリケーションを対象とした手法を提案した．2016 年 1 月に PHP7 がリリースされ，PHP7 では型推論や型不一致によるエラーを検出することができるようになった [35]．しかしながら，現在も PHP で開発されている多くの WEB サービスは PHP5 によって開発されており，全ての PHP アプリケーションが PHP7 に移行するには時間を要する [36]．よって本手法は，現状多くの PHP アプリケーションにおいて発生しうる，設定値の参照時型不一致ミスを検出するのに有益な手法である．

参考文献

[1] Raymond P.L. Buse, Westley R. Weimer: A Metric for Software Readability, *ISSTA '08*, pp. 121-130.

[2] History of PHP, Php Manual, http://php.net/manual/en/history.php.php

[3] Most popular server-side programming languages, https://w3techs.com.

[4] Anil Madhavapeddy, Richard Mortier, Ripduman Sohan, Thomas Gazagnaire, Steven Hand, Tim Deegan, Derek McAuley, Jon Crowcroft: Turning down the LAMP: software specialisation for the cloud, *HotCloud'10*, pp.11-19.

[5] M Sayagh and B Adams: Multi-layer Software Configuration: Empirical Study on Wordpress. *SCAM 2015*, pp.31-40 .

[6] Wordpress, https://wordpress.com

[7] D Oppenheimer, A Ganapathi, and D A. Patterson: Why do Internet services fail, and what can be done about it?, *USITS '03*.

[8] Y. J. Song, F. Junqueira, and B. Reed. Bft for the skeptics. In Proc. ACM SOSP' 09 Work in Progress Session.

[9] R. Johnson. More details on today' s outage. https: //www.facebook.com/notes/facebook-engineering/ more-details-on-todaysoutage/431441338919.

[10] Misconfiguration brings down entire .se domain in Sweden. http://www.circleid.com/posts/misconfiguration_brings_down_entire_se_domain_in_sweden

[11] Configuration error brings down the Azure cloud platform. http://www.evolven.com/blog/ configuration-error-brings-down-the-azure-cloud -platform.html.

[12] MongoDB Databases May Be Exposed by Security Misconfigurations, Security Intelligence, https://securityintelligence.com/news/mongodb-databases-may-be-exposed-by-security-misconfigurations/

[13] Bug in error reporting configuration, The Joomla! Forum, https://forum.joomla.org/viewtopic.php?t=708552

[14] Operators, PHP Manual, http://php.net/manual/en/language.operators.comparison.php

[15] Bug in the config file creation with automatic tool, WebChess, https://sourceforge.net/p/webchess/bugs/77/

[16] https://w3techs.com/technologies/details/pl-php/all/all

[17] Laleh Eshkevari, Fabien Dos Santos, James R. Cordy, Giuliano Antoniol: Are PHP applications ready for Hack?, *SANER'2015*, pp.63-72.

[18] Mona Attariyan, Jason Flinn: Using causality to diagnose configuration bugs, *ATC'08*, pp.281-286 .

[19] Zuoning Yin, Xiao Ma, Jing Zheng, Yuanyuan Zhou, Lakshmi N. Bairavasundaram, Shankar Pasupathy: An empirical study on configuration errors in commercial and open source systems, *SOSP '11*, pp. 159-172.

[20] Sai Zhang, Michael D. Ernst: Which configuration option should I change?, *ICSE2014*, pp.152-163.

[21] S. Nadi, T. Berger, C. Kastner, and K. Czarnecki: Mining configuration constraints: Static analyses and empirical results, *ICSE 2014*, pp. 140-151.

[22] X. Xia, D. Lo, W. Qiu, X. Wang, and B. Zhou: Automated configuration bug report prediction using text mining, *COMPSAC 2014*, pp. 107-116

[23] M. Attariyan and J. Flinn: Automating configuration troubleshooting with dynamic information flow analysis. *OSDI*, 2010.

[24] Y.-Y. Su, M. Attariyan, and J. Flinn: Improving configuration management with operating system causality analysis. *SOSP 2007*, pp.237-250 .

[25] Frances E. Allen: Control flow analysis, *SIGPLAN1970*, Volume5 Issue7 pp.1-19.

[26] Mark Weiser: Program slicing, *ICSE'81*, pp.439-449.

[27] Manu Sridharan, Stephen J. Fink, Rastislav Bodik: Thin slicing, *PLDI'07*, pp.112-122.

[28] 型, PHP Manual, http://php.net/manual/ja/language.types.php

[29] Hung Viet Nguyen, Christian Kastner, Tien N. Nguyen: Cross-language program slicing for dynamic web applications, *ESEC/FSE 2015*, pp.369-380.

[30] Hung Viet Nguyen, Christian Kastner, Tien N. Nguyen: Varis: IDE support for embedded client code in PHP web applications, *ICSE'15*, pp.693-696.

[31] Hung Viet Nguyen, Christian Kastner, Tien N. Nguyen: Varis: Building call graphs for embedded client-side code in dynamic web applications, *FSE 2014*, pp.518-529.

[32] VarAnalysis, GitHub, https://github.com/git1997/VarAnalysis

[33] mysql_connect function, PHP Manual, http://php.net/manual/en/function.mysql-connect.php

[34] mysql_select_db function, PHP Manual, http://php.net/manual/en/function.mysql-select-db.php

[35] PHP7, PHP Manual, http://php.net/manual/en/migration70.new-features.php

[36] Usage statistics and market share of PHP version 5 for websites, https://w3techs.com/technologies/details/pl-php/5/all

付録 A

Mis.Config のプログラム

プログラム A.1　Main クラス

```
1  public static void main(String[] args) {
2      for(String phpFilePath : entries){
3        String fullPhpFilePath = projectPath + '/' + phpFilePath;
4        Algorithm1 a1 = new Algorithm1();
5        HashMapPair stmtMap = a1.getMatchedStatements(phpFilePath);
6        HashMap<String, String> stmtsMap= stmtMap.getOldList();
7        HashMap<String,String> configList = Functions.getOptionsTypeList(
             CONFIG_PATH);
8        HashMap<String,String> trace = Functions.getTrace(fullPhpFilePath);
9        configList.forEach((option,originalType)->{
10         stmtsMap.forEach((k,v)->{
11           Pattern pattern = Pattern.compile(".+"+option+".+");
12           Matcher macher = pattern.matcher(v);
13           if(macher.find()){
14             List<String> affectedValues = getAffectedValuesList(option,
                   trace);
15             if(!isWellConfigured(option,originalType,v)){
16               result.append("-- "+ phpFilePath +"\n");
17               result.append(option+"\n");
18               result.append(originalType + ", "+v+"\n");
19               result.append("affectedValues:\n");
20               for(String o : affectedValues){
21                 result.append(o+"\n");}
22               result.append("\n\n");
23             }
24           }
25         });
26       });
27     }
28 }
```

プログラム A.2　getGraph

```
1 public static Graph getGraph(String projectPath, String entry) {
2     Graph cfg = new Graph();
3     ReferenceManager referenceManager = new ReferenceManager();
4     ReferenceManager refManager = new ReferenceDetector().detect(new File(
          projectPath, entry));
5     referenceManager.getDataFlowManager().addDataFlows(refManager.
          getDataFlowManager());
6
7     for (Reference reference : referenceManager.getReferenceList()) {
8       Position position = reference.getLocation().getStartPosition();
9       String line, statement = "";
10      try{
11        FileReader fr = new FileReader(position.getFilePath());
12        BufferedReader br = new BufferedReader(fr);
13        int count = 0;
14        while((line = br.readLine()) != null){
15          if(++count == position.getLine()) statement = line;
16        }
17        br.close();
18        fr.close();
19      }catch(IOException ex){
20        ex.printStackTrace();
21      }
22      cfg.addNode(new Node(reference.hashCode(),reference.getType(),
            reference.getName(),statement, position.getFileName()));
23    }
24    //Construct Edges
25    StringBuilder str = new StringBuilder();
26    str = new StringBuilder();
27    for (Reference ref1 : referenceManager.getReferenceList()) {
28      Position pos1 = ref1.getLocation().getStartPosition();
29      for (Reference ref2 : referenceManager.getDataFlowManager().
            getDataFlowFrom(ref1)) {
30        Position pos2 = ref2.getLocation().getStartPosition();
31        cfg.connectNode(cfg.getNodeFromIndex(ref1.hashCode()),cfg.
              getNodeFromIndex(ref2.hashCode()));
32      }
33    }
34    return cfg;
35 }
```

プログラム A.3　getOptionsTypeList

```
1 public static HashMap<String,String> getOptionsTypeList(String CONFIG_PATH
    ){
2   HashMap<String,String> optionsList = new HashMap<String,String>();
3   try{
4     String line;
5     BufferedReader br = new BufferedReader(new FileReader(CONFIG_PATH));
6     while((line = br.readLine()) != null){
7       Pattern pString = Pattern.compile("\\= [\"'].+[\"']");
8       Pattern pInt = Pattern.compile("\\= \\d+");
9       Pattern pBoolean = Pattern.compile("\\= (true|True|False|false)");
10      Pattern pattern = Pattern.compile("\\$\\w+?\\b");
11      Matcher m = pattern.matcher(line);
12      while(m.find()){
13        Matcher pS = pString.matcher(line);
14        while(pS.find()){
15          optionsList.put(m.group().replace("$", ""),"String");
16          continue;
17        }
18        Matcher pI = pInt.matcher(line);
19        while(pI.find()){
20          optionsList.put(m.group().replace("$", ""),"Int");
21          continue;
22        }
23        Matcher pB = pBoolean.matcher(line);
24        while(pB.find()){
25          optionsList.put(m.group().replace("$", ""),"Boolean");
26          continue;
27        }
28          //Exception
29      }
30    }
31  }catch(IOException e){
32    e.printStackTrace();
33  }
34  return optionsList;
35 }
```

プログラム A.4　getAffectedValuesList

```
1 public static List<String> getAffectedValuesList(String option, HashMap<
      String,String> SlicedList){
2   List<String> affectedValuesList = new ArrayList();
3   for(Entry<String, String> v : SlicedList.entrySet()){
4     String[] splitStatments = v.getValue().replace("&&","AND").split
          ("&",0);
5     for(String s : splitStatments){
6       if(s.contains("$"+option)) affectedValuesList.add(v.getKey());
7     }
8   }
9   return affectedValuesList;
10 }
```

プログラム A.5　isWellConfigured

```
1 public static Boolean isWellConfigured(String configName, String type,
      String stmt){
2     Boolean founded = false;
3     Pattern pattern_int1 = Pattern.compile(".+[-+*/]\\$"+configName+".+");
4     Pattern pattern_int2 = Pattern.compile(".+\\$"+configName+"[-+*/].+");
5     Pattern pattern_int3 = Pattern.compile("if(.+(\\>\\=.+|\\={2}\\d+))");
6     Pattern pattern_bool1 = Pattern.compile("if\\(!?\\$\\configName+(\\)
          |\\&{2}|\\|{2})");
7     Pattern pattern_bool2 = Pattern.compile("if(.+(\\={2}|\\!\\=)(true|
          false|True|False|TRUE|FALSE))");
8     Pattern pattern_string = Pattern.compile("mysql_connect\\(\\$\\
          configName+\\,?\\$\\configName+\\,?\\$\\configName+\\)");
9     Pattern pattern_insert = Pattern.compile("^\\$[0-9A-Za-z_\\[\\]']+=\\$
          "+configName);
10    Matcher macher_int1 = pattern_int1.matcher(stmt);
11    Matcher macher_int2 = pattern_int2.matcher(stmt);
12    Matcher macher_int3 = pattern_int3.matcher(stmt);
13    Matcher macher_bool1 = pattern_bool1.matcher(stmt);
14    Matcher macher_bool2 = pattern_bool2.matcher(stmt);
15    Matcher macher_string = pattern_string.matcher(stmt);
16    Matcher macher_insert = pattern_insert.matcher(stmt);
17
18    while(macher_int1.find() && !founded){
19      if(type == "Int")founded = true;    }
20    while(macher_int2.find() && !founded){
21      if(type == "Int")founded = true;    }
22    while(macher_int3.find() && !founded){
23      if(type == "Int")founded = true;    }
24    while(macher_bool1.find() && !founded){
25      if(type == "Boolean")founded = true;  }
26    while(macher_bool2.find() && !founded){
27      if(type == "Boolean")founded = true;  }
28    while(macher_string.find() && !founded){
29      if(type == "String")founded = true; }
30    while(macher_insert.find() && !founded){
31      founded = true;
32    return founded;
33 }
```

謝辞

本研究を行うにあたり，ご多忙の中，終始適切かつ丁寧なご指導をして下さった大須賀昭彦教授，田原康之准教授，清雄一助教に深く感謝致します．

大須賀昭彦教授は，3 年間の修士生活を通して，研究のみならず進路等について，非常に多くの貴重なご意見をいただきました．また，研究室・大学の枠を超えた活動の機会を与えてくださったことで，視野が広がり大きく成長することが出来ました．3 年間の丁寧なご指導，深く感謝申し上げます．

田原康之准教授（修士 2 年次），清雄一助教（修士 1 年次）はご多忙の中，週 1 回のゼミを始めとして熱心な研究指導を賜り，貴重な勉学の機会を与えてくださったことに深く御礼申し上げます．

トビタテ！留学 JAPAN 日本代表プログラムの 2 期生として，1 年間の海外活動に取り組んだ際，ご支援いただいた文部科学省・大須賀教授・庫川様をはじめとする本学留学生係のみなさまに感謝申し上げます．そして，3 年間共に励まし合い，研究や海外活動などの議論・意見を交換した，同期の皆様にも深く感謝申し上げます．

最後に，研究の機会と議論・研鑽の場を提供して頂き，ご指導頂いた国立情報学研究所/東京大学の本位田真一教授をはじめ活発な議論と貴重なご意見を頂いた研究グループの皆様，大須賀・田原研究室の皆様に感謝の意を表します．

研究業績

査読付き国内ワークショップ

1. **依田みなみ**, 清雄一，田原康之，大須賀昭彦: シンスライシングを用いた WEB アプリのための設定ファイル参照ミス検知手法の試作と評価, 10. 2016.（ショート発表, 貢献賞受賞）