平成２１年度　成果報告書

「静的及び動的解析の組み合わせによるWebアプリケーションの
セキュリティ診断システムに関する研究開発」

目　次

１　研究開発課題の背景

電子商取引を初めとしたWebサービスは企業から一般消費者までより日常的に利用されるようになった。その一方でインターネットに関連した事件事故は増加の一途を辿っている。昨今の景気低迷によりサービスの開発ベンダは競争が激化しており、より短期間に低コストでの開発を強く迫られており、劣悪な品質の開発物が多く実用されていることも事実である。このような状況が事件事故の増加の一因となり得ることは自明である。

ここ数年で特に新規Webサービスの開発案件に対して、運用前の検査として診断サービスを利用することはより一般的になったが、診断の品質は診断実施者依存という状況が続いており、単なる競争激化による低コスト化は進んでいるものの、未だ高価なサービスで有ることに違いは無い。

診断方式としては、ソースコード診断に関する案件も増加してはいるが、従来型の擬似攻撃診断を実施しているケースが圧倒的に多い。その原因として、ソースコードに関わる診断サービスは、より高コストであり且つここ数年間での技術進歩性が低いことが考えられる。より低コスト且つ正確な診断方式が求められていることは言うまでもない。

２　研究開発の全体計画

２－１　研究開発課題の概要

ソースコード診断がより一般的に実施されることに伴い、従来型の擬似攻撃診断とソースコード診断を併用するケースが増加している。この様なケースでは、それぞれの長所を生かして診断を実施してはいるものの、両技術の連携性はほぼ無いに等しい。

擬似攻撃診断はブラックボックス型の試験方式であり、入力に対する出力を解析することで脆弱性の存否を確認する。長所としては実環境での試験が可能であり、既知の攻撃パターンを流し込み、その出力に特定のパターンが出現しているかを確認すれば良く、試験の難易度は低い。短所としてはブラックボックス型の試験であるが故に診断結果は問題の内在性に留まり、問題の発生原因に関する詳細は知ることが出来ない。また試験実施時のコード網羅性も試験時の入力変数に依存し、どの程度の網羅性が得られているのかを知ることも出来ない。

ソースコード診断の擬似攻撃診断に対する優位点としては、ソースコードの網羅性が高く、問題発生時の原因箇所について指摘が可能である事があげられる。その一方でソースコード診断は入力データを伴わない静的解析であり、大量の誤検知発生や診断範囲がソースコード内に限られデータベースやその他システム構成に関わる問題については検知することが出来ない。

上記、２つの技術はそれぞれの長所が短所を上手く補う事になり、各診断を別々に行うことでも一定の効果は期待される。また、一方の診断（解析）結果を他方の診断実施時に反映させることで診断の品質向上が期待される。

本研究開発においては、アプリケーション開発段階においてソースコード解析を実施することで、その結果を擬似攻撃診断実施時に反映させ、誤検知の減少、網羅性の向上、また脆弱性発見時の問題箇所特定や絞り込みを可能とし、これら特徴を生かした修正支援も成果の対象として研究開発を実施する。

２－２　研究開発の最終目標（平成２２年９月末）

1. ソースコード診断ツールの研究開発
   （1）診断実施時のソースコード網羅率が80%ー90%を達成すること

本件研究開発ではソースコード網羅率が低くなりやすい擬似攻撃診断実施前に、予めソースコード診断を実施することで、擬似攻撃診断のパターン生成が内部構造を把握した状態で実行できる。これにより擬似攻撃診断実施時のソースコード網羅性を向上させることが可能となる。一般的なソースコード診断では 70%ー90%程度が網羅性の平均であり、特に Web アプリケーションの入出力に特化した静的解析では、80%ー90%程度の網羅率を確保できると考える。

（2）ソースコード診断における現状 50%の誤検知率を 30%以下に抑えること
ソースコード診断実施時に発生し易い誤検知を分析し、誤検知の発生頻度が高い問題に対しては、擬似攻撃診断実施時に検証することで、ソースコード診断実施時の誤検知を減少させる。

（3）擬似攻撃診断における現状 30%の誤検知率を 10%以下に抑えること
擬似攻撃診断実施時に判定しづらい問題点をソースコード診断により検証することで、相対的な誤検知発生率を減少させる。

2. 擬似攻撃診断と実行時内部トラッキングによる問題検知ツールの研究開発

（1）擬似攻撃診断によって検知された問題に対して制御フローを追跡可能なこと
本研究では擬似攻撃診断実施前に、ソースコードの各制御点に、実行時の追跡を可能とするデータ出力を行う為の関数ラッピングを行う。これにより、擬似攻撃診断を仕掛けた際の、各テストパターンに対してどのような制御フローが発生したのかを追跡することが可能となる。

（2）擬似攻撃診断によって検知された問題に対してデータフローを追跡可能なこと
本研究では擬似攻撃診断実施前に、ソースコード内の主要 API に対して動的に用意する検証用 API へと置換を行い、プログラム実行時にコールされた各 API への値出力を実施することで、各テストパターンに対してどのようなデータのながれが生じたのかを追跡することが可能となる。

3. 問題箇所の特定及び修正支援ツールの開発

（1）問題発生時の制御フロー、データフローを視覚化すること
本研究ではソースコード診断または擬似攻撃診断で検知された問題に対して、制御フロー及びデータフローの分析が可能となる。問題発生時の制御フローを視覚化することで問題箇所の特定を支援する。

（2）修正支援機能を提供すること
本研究では各種問題毎に一般的な修正案を示し、さらに上記制御フロー及びデータフローの結果を用いることで、どのような修正を実施すれば該当の問題を抑制できるのかといった修正案を示し、静的診断、動的診断の診断サイクルを繰り返す中で、問題の修正確認を行える。

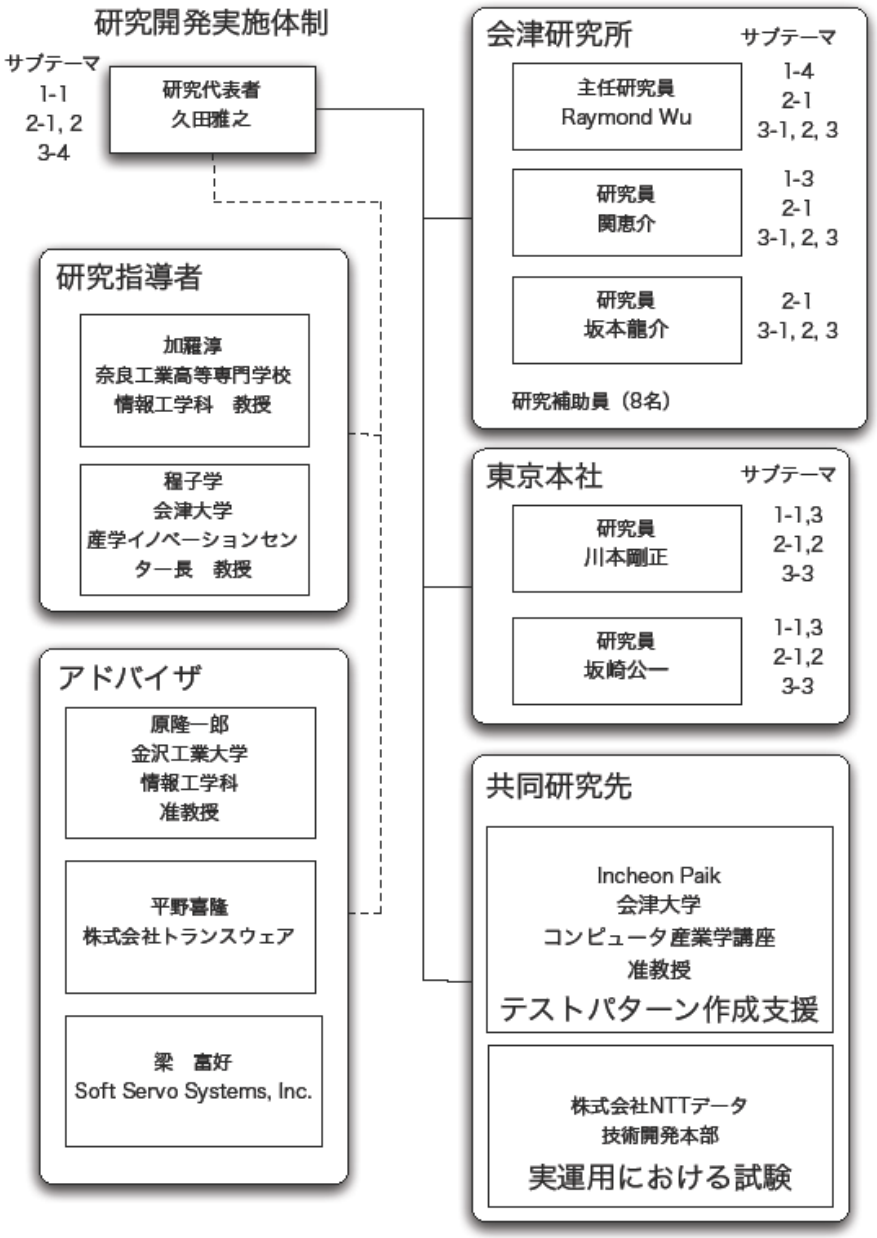
２－３　研究開発の年度別計画

金額は非公表

研究開発項目	20 年度	21 年度	22 年度	計	備　考
静的及び動的解析の組み合わせによる Web アプリケーションのセキュリティ診断システムに関する研究開発					
ア　ソースコードの診断技術に関わる研究	－ →	－	－	－	20 年度 10 月から 9 カ月
イ　ソースコード診断と擬似攻撃診断の相互連携手法の研究	－	－	－	－	21 年度 6 月から 6 カ月
ウ　擬似攻撃診断と実行時内部トラッキングによる問題検知手法の研究	－	－ →	－	－	21 年度 9 月から 1 年間
エ　問題箇所の特定及び修正支援に関わる研究	－	－	－	－	22 年度 4 月から 6 カ月
間接経費	－	－	－	－	
合　　計	－	－	－	－	

３　研究開発体制

３－１　研究開発実施体制



サブテーマ1:ソースコードの診断技術に関わる研究
　１－１:脆弱性検知を目的とした静的解析手法
　１－２:ソースコードの抽象化
　１－３:擬似攻撃実施時の網羅性調査
　１－４:脆弱性検知の多言語対応

サブテーマ2:ソースコード診断と擬似攻撃診断の相互連携手法の研究
　２－１:擬似攻撃診断実施時の網羅性向上
　２－２:擬似攻撃診断実施時の効果的テストパターン生成

サブテーマ3:擬似攻撃診断と実行時内部トラッキングによる問題検知手法の研究

３－１:トラッキング方法の確立・検証
３－２:トラッキングデータの解析法
３－３:脆弱性発生時の原因特定
３－４:擬似攻撃診断とソースコード診断の相互連携

研究補助員の分担:
検証用 Web アプリケーションの構築、実験データ収集
脆弱性データベースの調査、文献収集
プログラミングの補助業務
診断パターン調査、検証

４　研究開発実施状況

４－１　サブテーマ１：ソースコードの診断技術に関わる研究

４－１－１　脆弱性検知を目的とした静的解析手法

昨年度に引き続き PHP 言語で書かれたプログラムの解析モジュールの研究開発を実施した。今年度は下記サブテーマ２及びサブテーマ３の研究課題を実施する為に必要となる下記２つの機能実装を行った。

1. PHP 言語で書かれたプログラムから入力変数を自動抽出するモジュール
   擬似攻撃診断を実施する際には結果取得までに
   1.事前巡回処理
   2.テストケース生成
   3.診断実施（スキャン及びレスポンス解析）
   4.結果報告
   の４行程がある。

擬似攻撃診断の診断精度を表す指標としては、当該診断がソースコード全体のどの程度を網羅したのかというソースコード網羅性があげられる。診断実施時のソースコード網羅性が仮に３０％であれば、残りの７０％は診断がなされておらず、当該箇所にセキュリティ上の問題が含まれる可能性を否定できず、結果的に診断の品質は低いと言える。故に擬似攻撃診断における精度向上を目的としたソースコード網羅性について検証することにした。

オープンソースで開発されている Blog や CMS(WordPress, Simple PHP Blog)の Web アプリケーションを使って、巡回時ソースコード網羅性（本検証では分岐網羅を評価）を検証した。網羅性の計算には本研究開発におけるサブテーマ２で使用した当社独自の網羅性検証方式を利用し、プロトタイプ段階のツールを利用して網羅性を計算した。

WordPress に対して通常アクセス可能な範囲で３回の巡回を実施したところ、これら巡回時網羅性の平均は 30.75%であった。また Simple PHP Blog を同じ手順で巡回した際に得られた網羅性の平均は 19.60%であった。

アプリケーション名（バージョン）	巡回時網羅率（１回目）	巡回時網羅率（２回目）	巡回時網羅率（３回目）	巡回時網羅率（平均）
Simple PHP Blog (v0.5.1)	28.85%	31.69%	31.70%	30.75%
WordPress (v2.8.5)	20.68%	18.23%	19.91%	19.60%

網羅性検証において、網羅されなかった箇所をそれぞれ解析した結果、ある特定の変数値が入力されることによって当該箇所が実行されるケースが多くみられた。事前巡回の一般的なプロセスとしては、自動巡回及び手動巡回共に各ページ内に存在するリンク構造を元に遷移していくこととなり、当該ページを構成する HTML や JavaScript 内に含まれるリンク構造ではアプリケーション内部の全ての変数を網羅できておらず、アプリケーション内部に潜在的にある変数を網羅できないことが網羅性低下の一因であることを確認した。

巡回時に取得出来なかった各入力変数をソースコードの静的解析により抽出し、テスト生成のプロセスに反映させる事は網羅性の向上に繋がると考え、関連モジュールについて研究開発を行った。今回の実装では第一段階として、前記 Blog 及び CMS を対象とし、これらアプリケーションが PHP で構成されていることから、PHP で書かれたアプリケーションの入力変数を自動抽出するモジュールを開発することとなった。なお PHP の言語仕様では入力変数は_GET 及び_POST によってソースコード内で参照される為、当該モジュールはソースコードを解析しこれら入力変数が参照される箇所を検出、使用されている変数名を抽

出する動作となる。
本モジュールの動作としては、予め指定した予約語について変数名（今回は_GET 及び_POST を対象とした）を自動で取り出し記録する。_GET 及び_POST は予約語で有る為、各ソースコードに対して、文字列前方一致で当該予約語の使用を検索する方法を使用し、実装を完了した。当該モジュールはサブテーマ２の事前処理として利用された。

2. PHP 言語で書かれたプログラムにログ命令を自動追加するモジュール
サブテーマ３では、擬似攻撃診断実施時に発見された脆弱性が発現した制御フローを特定、明示出来ることを目標としており、当該目的達成の為に必要となる Web アプリケーション実行時の各制御フローを記録する為の方法について研究開発を実施した。

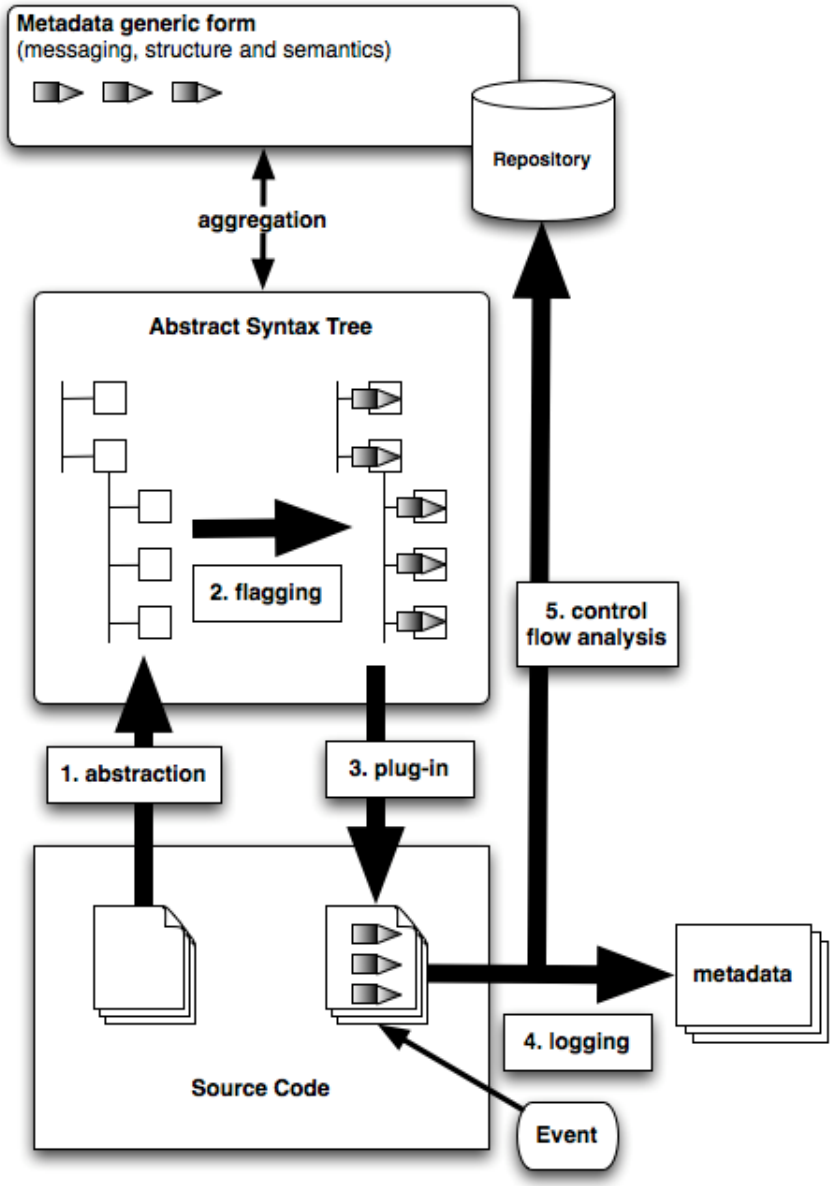


図 1
制御フロー特定を目的としたソースコード改変技術

図１は本研究開発の概要を示すもので、基本的な考え方としては診断対象となるアプリケーションのソースコードから、診断用途のソースコードを生成し、当該生成コードを用いてプログラムを実行すると内部状態が記録され、結果として制御フローを特定で

きるというものである。
1.診断対象のソースコードは、字句解析及び構文解析により抽象構文木へと変換される（abstraction）
2.抽象構文木は解析され、if 文や switch 文など制御を決める命令付近に記録を目的とした特殊なコードを埋め込む（flagging）
3.変更された抽象構文木は、ソースコードに書き戻され（plug-in）実行される。
4.実行時にプログラムに対して何らかのイベントが発生した際には、各制御文においてメタデータが記録される（logging）
5.これらメタデータの記録から、当該イベントに対する実行フローは解析により特定される（control analysis）
本研究開発では、実装期間の短縮の為に抽象構文木の生成には Eclipse に含まれる PDT を利用した。

以上の様にソースコードの改変技術をベースとした制御フローの特定技術を確立させ、擬似攻撃診断実施時の制御フロー特定及び網羅性の評価を実現する技術確立に成功した。

なお、PDT と同様の JDT による抽象構文木生成、抽象構文木の書き換え及びソースコードの書き戻しも試験し、動作確認を行った。本ツールは Java ベースのアプリケーション(Servlet, Struts を用いたアプリケーション)にも対応を完了している。

４－１－２　擬似攻撃実施時の網羅性調査

前記ログ命令の自動追加モジュールを利用することで、ソースコード内の各分岐点に通過条件を記録するためのコード改変が可能となった。各分岐点の通過可否を判定することで、プログラム実行時の分岐網羅性を評価することが可能となる。Web アプリケーションが実行された際のメタ情報は個別識別子により管理され、実行時の識別子は当該アプリケーションが返したレスポンス内部に埋め込まれる。以下は HTML のサンプルであり、HTML の閉じタグ直前に埋められた数字が上記識別子となる。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>NST(デモサイト)</title>
</head>
<body>
...省略...
</body>
<!--_NST_SCA_CTRL_72,35,79,92,77,90,75,89,71,73,80,-->
</html>
```

識別子を順次参照していくことで、この HTML が生成された際の制御フローは以下の通り特定される。

```
72 | _NST_SCA_CTRL_ | footer.php | TopStatement | 0 | 0
35 | _NST_SCA_CTRL_ | content/welcome.php | TopStatement | 0 | 0
79 | _NST_SCA_CTRL_ | index.php | block | 29 | 0
92 | _NST_SCA_CTRL_ | menu/public_left_menu.php | TopStatement | 0 | 0
77 | _NST_SCA_CTRL_ | index.php | block | 20 | 0
90 | _NST_SCA_CTRL_ | main_navi/public_main_navi.php | TopStatement | 0 | 0
75 | _NST_SCA_CTRL_ | index.php | block | 8 | 0
89 | _NST_SCA_CTRL_ | lib.php | TopStatement | 0 | 0
71 | _NST_SCA_CTRL_ | do.php | TopStatement | 0 | 0
73 | _NST_SCA_CTRL_ | header.php | TopStatement | 0 | 0
80 | _NST_SCA_CTRL_ | index.php | TopStatement | 0 | 0
```

オープンソースを中心に上記コード改変技術を用いて、分岐網羅性の評価を実施したところ、当初想定していた以上に分岐網羅性の確保が難しいことがわかった。一般的なクローリングアルゴリズムを用いて自動巡回した場合は最大で約３０％、人が無意識に巡回した場合で最大で約５０％、相当意識して巡回作業を実施しても最大で約７０％程度の網羅性しか確保できない結果となった。なお大規模なアプリケーションほど網羅性は低く、網羅性が３０％に達しないケースも多かった。

高網羅の条件としては前記の通りソースコードの分岐条件に使われている変数が、ページ内に存在するリンクの URL にて変数として参照されていることが必要である。該当の変数が呼ばれていない場合は、常に一定の分岐条件を満たす事から網羅性が低くなることがわかった。本結果はサブテーマ２における網羅性向上のアプローチを導き出した。

４－１－３　脆弱性検知の多言語対応

昨年度の成果を基に、PDT 及び JDT が生成する抽象構文木を統一して解析可能なユニバーサルアダプタの設計及び実装を行った（図２）。言語仕様の差異は抽象構文木を用いることで吸収され、プログラムの変更手続きは構文木上の操作に変換される。また言語単位で抽象構文木の定義情報が異なる場合には、各ノードの対応関係を別に定義しておくことで、前記コード解析エンジンの多言語対応が容易になった。本研究開発で作成したソースコード解析エンジンは PHP 言語及び Java 言語に対応している。

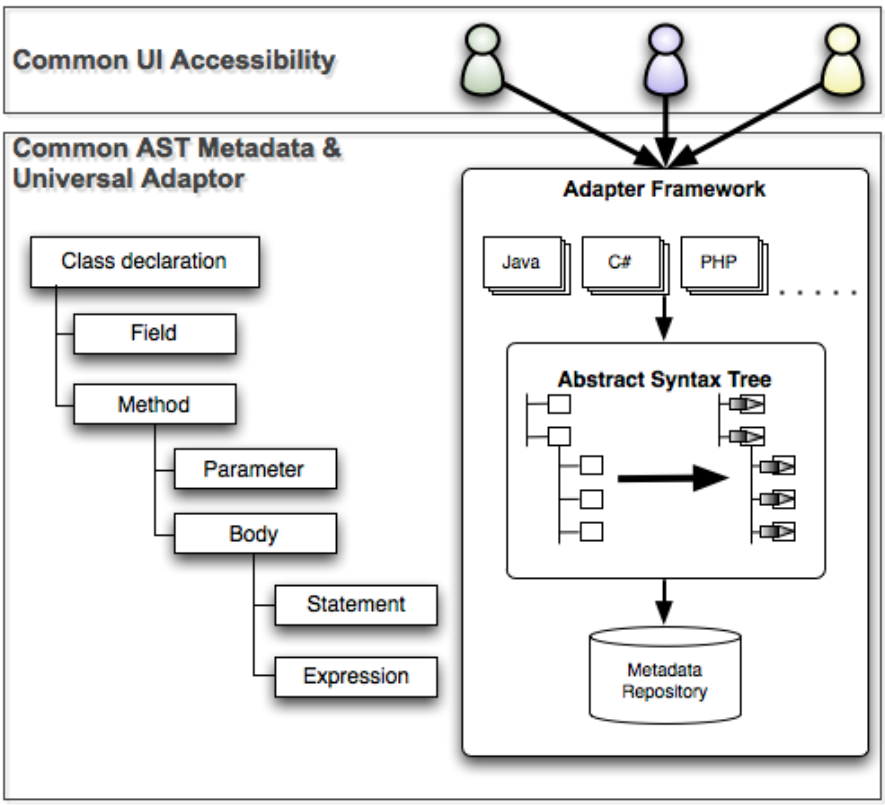


図 2
ソースコード改変の多言語対応

４－２　サブテーマ２：ソースコード診断と擬似攻撃診断の相互連携手法の研究

４－２－１　擬似攻撃診断実施時の網羅性向上

主要な Web アプリケーションサーバーの仕様を調査し、それぞれの動作を検証した結果、リクエストに含まれる変数項目は事前にパースされ、コード実行時に参照があった場合に値を返す仕組みが取られており、コード内で参照されていない変数を URL に付加しても特に影響が無い事がわかった。PHP 及び Java の主要な環境で検証を実施し、問題が発生しな

いことを確認した。
本結果を踏まえ、事前にソースコード解析を実施し、入力可能な変数項目を抽出しておくことで、巡回時に不足している入力変数を自動追加し網羅性を向上させる為の方法を考案し、下記テストパターン生成の手法確立に至った。

４－２－２　擬似攻撃診断実施時の効果的テストパターン生成

前記、変数抽出モジュールを利用して、擬似攻撃診断実施時の網羅性向上を目的とした連携手法について研究を実施した。
擬似攻撃診断は、実際に攻撃を仕掛けてその反応を調べる診断方式である。診断時には事前巡回を実施し、巡回時に利用された URL や変数項目に対して既知の攻撃パターンを総当たりで埋め込んでテストケースを生成する。サーバーへ実際に各テストケース（リクエスト）を送信し、そのレスポンスを解析、問題の有無を判定する。
前述の調査結果から、発生したリクエストに含まれる変数不足が網羅性の低下原因として多く、ソースコード内で参照される変数項目を網羅することで分岐網羅性の向上が期待されることから、結果的に変数追加は擬似攻撃診断で使用されるテストケースを効果的に生成する作用を持つ。
今回は、Web ページ内からは参照されない変数をソースコード内部で定義し試験を実施した。当該検証アプリケーションは弊社内で試験用に作成したものであり、デバッグ用途の変数が内部に５つ存在、これら変数により新しく分岐が発生するよう構成されている。
当該検証アプリケーションにおける分岐点は９３カ所であり、巡回時の分岐通過点は４６カ所で巡回時分岐網羅率は約５０％であった。これら巡回データに対してソースコードから抽出された前記５つの変数を追加することで、通過点は５６カ所となり分岐網羅率は約６０％まで向上した。結果的に追加された変数により１０カ所の分岐が新たに発生したこととなる。
また WordPress 管理画面のログイン関連ページに対して本手法を適用したところ、当該 URL の分岐網羅率は 6.54%であったが、内部から抽出された変数の追加により分岐網羅率は 12.77%まで向上した。これにより一般的に使用されているアプリケーションに対しても本手法の有効性が確認された。
今回の手法適用による分岐網羅率の向上はアプリケーションにより大きく異なる。しかしながら Web ページ内で参照されない変数により制御が変わるアプリケーションでは、確実に分岐網羅性を向上させることが可能となる。本成果については、一部技術の特許申請を予定しており、研究成果については平成 22 年度中に論文投稿を予定している。

４－３　サブテーマ３：擬似攻撃診断と実行時内部トラッキングによる問題検知手法の研究

４－３－１　トラッキング方法の確立・検証

擬似攻撃診断実施時の制御フローを特定する技術について研究を実施した。昨年度の成果及び今年度前期の成果により、PHP 言語及び Java 言語で書かれたプログラムに対して、各種制御命令の成立条件をプログラム実行時に記録出来るコード改変を安定的に行う技術を確立した（図１）。
対象のメタデータとして、プログラムコードの ID、行番号、制御命令の種類、コードインデントの深度を記録し、擬似攻撃診断の各テストケースに対してそれぞれどのような制御を取ったのか出力が可能となった（4- 1- 2にて例示)。また各制御文の正否を特定できることから、分岐網羅性の評価もでき、改変コードに対する事前巡回及び擬似攻撃診断の実施時には分岐網羅性を評価することが可能となった。なお、本研究における分岐網羅性評価は全て本成果を用いて取得している。
現時点では、各分岐文の直後にメタデータを記録する為の命令埋め込みを行っているが、脆弱性が実際に発現している箇所が直前の分岐命令から離れている場合には、該当行を見

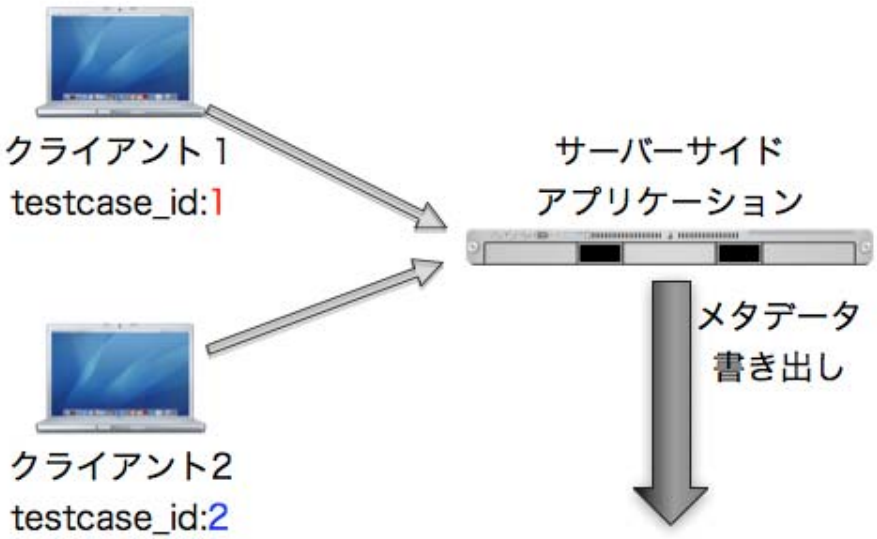
つける為に相当のコードを目視する作業が必要となる。本研究課題は平成 22 年度も継続して実施する予定であり、発生した脆弱性をより特定しやすくする為のトレース技術を重点的に実施する予定である。

本成果の関連技術については国際会議 2 件、学術雑誌 1 件が発表済みであり、また PCT による国際出願 1 件も完了しており、該当の出願は日本国への国内移行済みである。

４－３－２　トラッキングデータの解析法

トラッキングデータはプログラム実行時に自動的に記録されるが、擬似攻撃診断実施時には複数のスレッドが同時にテストケースを送信することが一般的であり、各テストケースと記録されたトラッキングデータを効率的に対応づける仕組みが必要となる。

本研究開発ツールでは、PHP に対する診断実施時には各テストケース送信時にリクエストヘッダにテストケースの識別番号を埋め込み、コード改変により埋め込まれたロギング命令が該当の識別番号をログとしてメタデータと共に記録することで、上記複数スレッドによるテスト送信に対応した仕様となっている(図３)。




```
72|_NST_SCA_CTRL_| footer.php | TopStatement | 0 | 1
35|_NST_SCA_CTRL_| content/welcome.php | TopStatement | 0 | 1
79|_NST_SCA_CTRL_| index.php | block | 29 | 1
72|_NST_SCA_CTRL_| footer.php | TopStatement | 0 | 2
35|_NST_SCA_CTRL_| content/welcome.php | TopStatement | 0 | 2
79|_NST_SCA_CTRL_| index.php | block | 29 | 2
92|_NST_SCA_CTRL_| menu/public_left_menu.php | TopStatement | 0 | 1
92|_NST_SCA_CTRL_| menu/public_left_menu.php | TopStatement | 0 | 2
77|_NST_SCA_CTRL_| index.php | block | 20 | 1
77|_NST_SCA_CTRL_| index.php | block | 20 | 2
90|_NST_SCA_CTRL_| main_navi/public_main_navi.php | TopStatement | 0 | 1
75|_NST_SCA_CTRL_| index.php | block | 8 | 1
89|_NST_SCA_CTRL_| lib.php | TopStatement | 0 | 1
90|_NST_SCA_CTRL_| main_navi/public_main_navi.php | TopStatement | 0 | 2
75|_NST_SCA_CTRL_| index.php | block | 8 | 2
89|_NST_SCA_CTRL_| lib.php | TopStatement | 0 | 2
71|_NST_SCA_CTRL_| do.php | TopStatement | 0 | 1
73|_NST_SCA_CTRL_| header.php | TopStatement | 0 | 1
80|_NST_SCA_CTRL_| index.php | TopStatement | 0 | 1
71|_NST_SCA_CTRL_| do.php | TopStatement | 0 | 2
73|_NST_SCA_CTRL_| header.php | TopStatement | 0 | 2
80|_NST_SCA_CTRL_| index.php | TopStatement | 0 | 2
```

図 3

複数スレッドからのテスト送信とメタデータの記録識別

４－４　総括

本報告書作成時点で、研究開発における計画の遅延は発生しておらず、順調に進んでいる。

サブテーマ１：ソースコードの診断技術に関わる研究

昨年度は主に PHP 言語で書かれた Web アプリケーションのソースコードに関わる技術を研究開発し、また解析エンジンを言語非依存にする為に、ソースコードプログラムを抽象化し解析する方法について研究を実施した。これにより PHP と Java についてそれぞれ抽象構文木を構成し、抽象構文木上で解析を実施することにより言語非依存の解析エンジンを構築することに成功した。今年度は昨年度に引き続き PHP 言語で書かれたプログラムの解析モジュールについて研究開発を実施し、サブテーマ２及びサブテーマ３の研究課題を実施する為に必要となる、PHP 言語で書かれたプログラムから入力変数を自動抽出するモジュールと PHP 言語で書かれたプログラムに対してログ命令を自動追加するモジュールの２つの機能実装を行った。サブテーマ１全体の成果としては、

１．言語非依存の解析エンジン構築法（昨年度及び今年度実績）

２．Eclipse フレームワークの PDT 及び JDT を用いた言語解析及びコード改変に関わる技術確立（昨年度及び今年度実績）

３．抽象構文木間の差異を吸収するための抽象化（昨年度実績）

上記３点があげられる。またこれら成果は今年度取り組んだサブテーマ２及びサブテーマ３の基礎技術として用いられた。

サブテーマ２：ソースコード解析から擬似攻撃診断の網羅性を向上させる

サブテーマ３：擬似攻撃診断を実施した際に各リクエストに対する制御フローを特定する

サブテーマ２の網羅性向上については事前に想定していた数値よりも一般的に得られる網羅性が低いことがわかった。目標数値はインターネット上の情報や各種文献を参照して設定していたが、実際に実験を開始してみると Web サイト巡回時の平均的な網羅性は５０％に満たない事が多く、当然ながら対象となるアプリケーションにより大きく異なる結果となった。

現時点での本研究開発ツールにおける平均的な網羅性は６０％程度であり、最終目標値である８０％ー９０％には差が生じている。本サブテーマについては平成 21 年度で完了となるが、次年度においてもサブテーマ３と平行して検証を継続し、少しでも最終目標値に近づけるよう調整を行う予定である。

サブテーマ２に関わる成果発表は次年度に持ち越しとなるが、サブテーマ３に関わる成果として学術雑誌、国際会議に数件採択、発表が完了しており、国際的に認められる成果が得られた。

５　参考資料

５－１　研究発表・講演等一覧

＜研究論文＞

(1)雑誌名：Journal of Object Technology
　発表者：Raymond Wu and Masayuki Hisada
　発表タイトル：SOA Web Security and Applications
　発表月日：平成 22 年 3 月

(2)雑誌名：International Journal of Computer Science and Network Security
　発表者：Raymond Wu and Masayuki Hisada
　発表タイトル：The Architectural Review of Web Security in Static and Dynamic Analysis

発表月日：平成 21 年 9 月

＜外国発表予稿等＞

(1)会議名：5th International Conference on Global Security, Safety, and Sustainability, ICGS3’09
発表者：Raymond Wu and Masayuki Hisada
発表タイトル：Static and Dynamic Analysis for Web Security in Generic Format
発表月日：平成 21 年 9 月

(2)会議名：The 2009 International Conference on Internet Computing, ICOMP’09
発表者：Raymond Wu and Masayuki Hisada
発表タイトル：Static Analysis for Web Security in Abstract Syntax Format
発表月日：平成 21 年 7 月

５－２　産業財産権

５－２－１　出願特許数

国内出願　　１件、　国際出願　　１件

５－２－２　公開特許一覧

該当なし

５－２－３　登録特許一覧

該当なし