Vol.2013-DPS-154 No.27
Vol.2013-CSEC-60 No.27
2013/3/14

静的解析を用いたWebアプリケーション脆弱性対策方法

植田武[†1]　桜井鐘治[†1]

Webアプリケーションの脆弱性を防ぐには，アプリケーション内部で行う処理内容に応じて，適切な脆弱性対策を実装する必要がある．Webアプリケーションの脆弱性対策は，Webアプリケーションの内部で行われる処理内容によりその方法が異なるため，正しい対策を行うには，開発者の脆弱性に対する知識やスキルが必要である．近年，この問題を解決するために，Webアプリケーションの脆弱性対策のためにガイドラインの策定や，対策方法を実装したセキュアライブラリの開発が行われてきた．しかしながら，依然として，セキュアライブラリを適切に利用するためには，開発者に知識やスキルが必要であることに変わりはなく，また，知識やスキルがあったとしても開発者のミスによって対策が抜けてしまうという可能性もある．そこで本稿では，上記の問題を解決するために，静的解析の結果を用いて，脆弱性対策を行うセキュアライブラリの選択を動的に行う方法について検討した．本稿では，検討した方式について説明する．

Web Application Vulnerability Measure Method using Static Analysis

TAKESHI UEDA[†1]　SHOJI SAKURAI[†1]

In order to prevent the Web application vulnerability, according to the contents of processing performed inside application, it is necessary to mount a suitable vulnerability measure. Since the method changes with contents of processing performed inside Web application, in order to perform the right measure, a developer's knowledge and skill are required for the vulnerability measure against Web application. In recent years, in order to solve these problems, decision of the guideline for the vulnerability measure against Web application and development of the secure library which mounted the measure method have been performed. However, in order to use a secure library appropriately, even if there is no change in knowledge and skill being required for a developer and there are knowledge and skill, there is a possibility that a measure will fall out by a developer's mistake. So, in this paper, in order to solve the above-mentioned problem, the result of static analysis was used and how to choose dynamically the secure library which performs a brittle measure was examined. This paper explains the examined system.

1. はじめに

従来からOSやアプリケーションに存在するセキュリティ上の脆弱性を攻撃することで，データやプログラムの改ざん，コンピュータやアプリケーションの強制停止，コンピュータへの不正侵入や不正操作等の被害が発生している．近年では，インターネットに公開するWebアプリケーションへの攻撃が増加しており，その攻撃の目的も攻撃者自身のスキルを誇示する愉快犯的な行為から，個人情報や企業情報の採取による金銭目的の犯罪や，特定企業や政府機関を狙った事件へと推移している[1]．また，Webサイトの改ざんによるウイルス配布サイトへのリンクの埋め込み等により，Webサイトの利用者に被害が及ぶこともある．また，Webアプリケーションが複雑大規模化するなか，Webアプリケーションの脆弱性が多数報告されている[2]．

Webアプリケーションの脆弱性は，アプリケーション内部で行う処理内容に応じて，脆弱性の種類が異なり，脆弱性の種類毎に異なる対策を実装する必要がある．正しい対策を行うためには，開発者に脆弱性が発生する仕組みに対する知識やスキルが必要である．例えば，データベースの操作を行うSQL文を動的に生成する場合に発生するSQLインジェクションを防ぐには，SQL文に含める入力データからSQLの構文上の意味を変えてしまう文字列を変換する等の無害化を行う．また，HTML処理を動的に生成する処理で発生するXSS（Cross Site Scripting）を防ぐには，HTMLに含める入力データからスクリプトとして動作する文字列を変換する等の無害化を行う．さらにXSSでは，入力データを含めるHTMLの構造によって無害化を行う方法が異なる．このように，データベースの操作，HTMLの出力等のWebアプリケーション内部で行われる処理内容や，外部からの入力データを利用するHTMLの構造等の各処理で利用するデータによってWebアプリケーションの脆弱性対策は異なるため，脆弱性の対策を適切に行うためには，開発者に脆弱性に対する十分な知識やスキルが必要である．

これらの問題を解決するために，Webアプリケーションの脆弱性対策を行うためのガイドラインの策定や，対策方法を実装したセキュアライブラリの開発が行われてきた．しかし，実際にセキュアライブラリをWebアプリケーションに適用する場合にでも，脆弱性が発生する仕組みを開発者が理解した上で，適切な脆弱性対策を選択し，適切な箇所で脆弱性対策を行う必要があり，依然として，開発者に脆弱性に対する知識やスキルが必要であることに変わりはない．また，脆弱性に対する知識やスキルがあったとしても開発者のミスによって脆弱性対策が抜けてしまうという可能性もある．

筆者らは，上記の問題を解決するために，静的解析の結

†1 三菱電機株式会社
Mitsubishi Electric Corporation

Vol.2013-DPS-154 No.27
Vol.2013-CSEC-60 No.27
2013/3/14

果を用いて，脆弱性対策を行うセキュアライブラリの選択を動的に行う方法について検討した．本稿では，検討した方式を説明する．

以下，2 章でセキュアな Web アプリケーションを実現するための関連技術を紹介し，3 章で提案方式の詳細を述べる．4 章で提案方式について考察を述べ，5 章で本稿をまとめる．

2. 関連技術

セキュアな Web アプリケーションを実現するための関連技術について簡単に紹介する．

2.1 セキュアプログラミング

プログラム自体に脆弱性を作りこまないようにするための設計方法やプログラミング方法である．Web アプリケーションに対するセキュアプログラミングは, OWASP（The Open Web Application Service Project），IPA（Information-technology Promotion Agency, Japan：独立行政法人 情報処理推進機構）等によって，表 1 に示すようなガイドラインが公開されている．

表 1　セキュアプログラミングに関する主なガイドライン

名称	発行元	概要
OWASP Top 10 [3]	OWASP	Web アプリケーションの脆弱性トップ 10 について，脆弱性の概要，確認方法，攻撃シナリオ例と防止方法を説明.
OWASP Code Review Project V1.1 [4]	OWASP	ソースコードのレビューするためのガイドライン．レビューを行う際に調査すべき機能，問題となるコードの例，良いコードの例が記載.
PHP To 5[5]	OWASP	「SANS Top 20 2005's PHP action」に基づいて分析した PHP の脆弱性トップ 5 の防御方法が記載.
IPA セキュア・プログラミング：講座 Web アプリケーション編[6]	IPA	セキュアな Web アプリケーションを構築するためのガイドライン．主に設計レベルでの対策方法を解説.
安全なウェブサイトの作り方[7]	IPA	セキュアな Web アプリケーションを構築するためのガイドライン．IPA に報告される主な脆弱性を作りこまないために実装時に気をつけるポイントを解説.
安全な SQL の呼び出し方[8]	IPA	安全なウェブサイトの作り方の別冊．SQL インジェクションの対策方法について，具体的なサンプルコードを示して解説.
The WASC Threat Classification v2.0[9]	WASC	Web アプリケーションの各脆弱性についての説明，攻撃方法や脆弱性のあるコードの例及び対策方法の概要が記載.
CWE/SANS Top 25 [10]	CWE/ SANS	脆弱性の原因となる危険度の高いプログラミングエラーのトップ 25 個．各脆弱性の詳細，悪いコード例，検出方法が記載.

2.2 セキュアライブラリ

Web アプリケーションの脆弱性の対策方法を実装したライブラリである．このライブラリを利用することで，対策方法が異なる Web アプリケーションの各脆弱性対策について，実装誤りを防ぐことができる．OWASP が公開している ESAPI[11]や，Microsoft の Anti-XSS Library[12]等が知られている．

2.3 静的解析技術

プログラムの解析手法の一つであり，プログラムを実行することなく解析するための技術である．本技術を使うことで，ソースコードがコーディング規約等に準拠しているかを検査することができる．

Web アプリケーションに対する静的解析技術に関する研究開発としては，Web アプリケーション内部で生成される SQL 文や HTML が, 外部からの入力データによって壊される可能性があるかどうかや，ソースコード内の文字列データの流れを追跡して変数の値がどのような値（文字列パターン）となるかを解析する文字列解析（String Analysis）[13][14][15]や，外部から入力された「汚染」されたデータが脆弱性を発生する可能性がある関数（HTML を出力する関数やデータベース操作を行うための関数）の引数として到達しないかを検証する汚染解析（Taint Analysis）[16]等が知られている．

2.4 Web アプリケーション診断技術

Web アプリケーションの脆弱性の有無を調査する技術である．この技術では，実際に Web アプリケーションを動作させて擬似攻撃データを Web アプリケーションに入力し，Web アプリケーションからの応答を解析することで脆弱性の有無を判定する．

2.5 WAF

保護対象とする Web アプリケーションへの入力を監視し，不正な入力があった場合，管理者への通報や，通信の遮断等の対策を実行するフィルタリング技術･装置である．類似する技術として IDS（Intrusion Detection System）や，IPS（Intrusion Prevention System）が知られているが，IDS や IPS はネットワーク層（あるいはトランスポート層）で動作するため，クライアントと Web アプリケーションとの通信については限定的な解析能力しか無い．これに対し，WAF はアプリケーション層上で動作し，HTTP やクッキーといった Web アプリケーションが利用する通信プロトコルをすべて解析することができる．

2.6 既存研究の課題

開発した Web アプリケーションから脆弱性を根本的になくすには，セキュアプログラミングを行う必要がある．

Vol.2013-DPS-154 No.27
Vol.2013-CSEC-60 No.27
2013/3/14

セキュアプログラミンを行う場合に,実装の手間の削減や,実装誤りによる脆弱性の混入を少なくするためには，検証されたセキュアライブラリを利用することが有効である．

しかしながら，開発者が必要なセキュアライブラリを有効的に活用するためには，脆弱性が発生する仕組みを理解した上で，適切な脆弱性対策を選択し，適切な箇所で脆弱性対策を行う必要があり，開発者の脆弱性に対する知識が不足している場合には，対策抜けや誤った対策を選択してしまう可能性がある．選択した対策が正しい場合にでも，対策が重複する等のミスにより，対策が入力データに悪影響を及ぼすこともある．なお，セキュアライブラリの適用を簡単にするために，開発者が記載した設定ファイルに従って，リクエストの受信時に脆弱性対策の処理を一括して適用するセキュアライブラリも存在する．この方式では，対策を実行する箇所を調べる必要もなく，対策が重複するミスを防止できるが，リクエスト受信時に選択された対策を一括して行うため,複数の脆弱性対策が必要な場合には,ある脆弱性対策で無害化対象となった文字が利用できない等の他の処理に悪影響を及ぼすという可能性がある．

3. 提案方式

3.1 コンセプト

従来技術の課題を解決する方法として，Web アプリの運用開始前に，プログラムを静的解析して得られたデータや処理の流れに基づいて，Web アプリの実行環境で脆弱性対策を動的に行う方法を提案する(図 1 を参照).

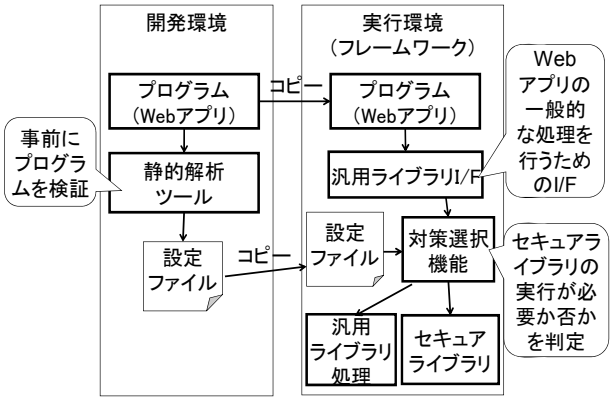

図 1　提案方式の概念図

3.2 機能構成

提案方法を実現するための機能構成を図 2 に示す.

開発環境は，開発者が Web アプリケーションを実装・試験するための環境である．実行環境は，運用を開始した Web アプリケーションを実行するための環境である.

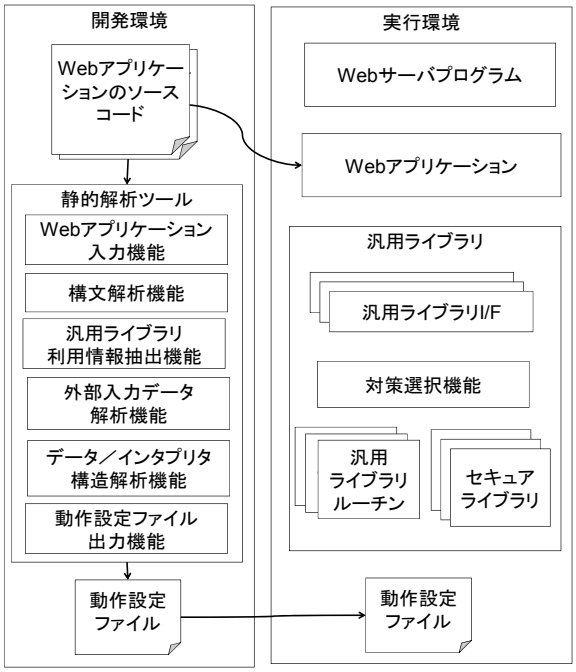

図 2　自動選択方式を実現するための機能構成

(1) 開発環境の機能構成

開発環境は，静的解析ツールを備える．静的解析ツールは, 読込んだ Web アプリケーションのソースコードを静的に解析し，HTML 出力処理やデータベースの操作等の Web アプリケーションの一般的な処理を行うための汎用ライブラリの動作設定ファイルを出力する.

静的解析ツールは，以下の機能から構成される.

- Web アプリケーション入力機能
  Web アプリケーションのソースコードを読込む機能.
- 構文解析機能
  読込んだ Web アプリケーションのソースコードを, ソースコードのプログラム言語の構文に従って解析し, 静的解析ツール内で解析を行うための抽象構文木を作成する機能.
- 汎用ライブラリ利用情報抽出機能
  構文解析機能が生成した抽象構文木を調査して, 汎用ライブラリの利用に関する情報として, 汎用ライブラリの呼出しを行うソースコードの位置や, 呼出される汎用ライブラリ I/F の種類等を抽出する機能.
- 入力データ解析機能
  構文解析機能が作成した抽象構文木を調査して, 汎用ライブラリの呼出しに利用されている入力データのデータフロー解析や, その入力データの取りうる値の範囲を解析する機能.
- データ／インタプリタ構造解析機能
  HTML 文や SQL 文等の Web アプリケーションの内部で利用するデータやインタプリタの構造を解析する機能.

- 動作設定ファイル出力機能
  汎用ライブラリ利用情報抽出機能，入力データ解析機能，データ／インタプリタ構造解析機能の結果に基づいて，汎用ライブラリが実行時の動作を決定するための動作設定ファイルを出力する機能．

(2) 実行環境の機能構成

実行環境は以下から構成される．

- Web サーバプログラム
  Web アプリケーションを実行するためのプログラム．
- Web アプリケーション
  実行環境上で稼働する Web アプリケーション．Web アプリケーションのソースコードは，開発環境の静的解析ツールを用いて事前に解析されているとする．
- 汎用ライブラリ
  HTML 出力処理，データベースの操作，ファイルの読み書き等の Web アプリケーションで行われる一般的な処理を行うための機能を提供するライブラリ．汎用ライブラリは以下の詳細機能で構成される．
  - 汎用ライブラリ I/F
    Web アプリケーションから汎用ライブラリを利用するための API（HTML 出力処理やデータベース操作等の処理の分類ごとに I/F を用意）
  - 対策選択機能
    動作設定ファイルを読み込み汎用ライブラリ内部で脆弱性の対策が必要か否かを判定するための機能
  - セキュアライブラリ
    Web アプリケーションの脆弱性に対する対策処理を行うためのライブラリ（脆弱性の対策ごとに複数用意される）
  - 動作設定ファイル
    静的解析ツールで解析した結果が含まれるファイル

3.3 処理フロー

(1) 開発環境での処理フロー

開発環境での処理フローを図 3 に示す．

① Web アプリケーション入出力機能において，Web アプリケーションのソースコードを読込む．

② 構文解析機能において，①で読込んだ Web アプリケーションのソースコードを，プログラム言語の構文に従って解析し，ソースファイルの単位で抽象構文木を作成する．

③ 汎用ライブラリ利用情報抽出機能において，②で作成した抽象構文木をそれぞれ解析して，Web アプリケーションが汎用ライブラリ I/F を呼出している命令文を特定し，汎用ライブラリ I/F を呼出す可能性

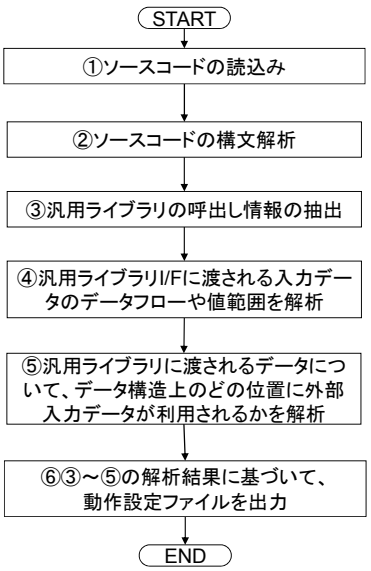

図 3　開発環境での処理フロー

がある命令文の情報（汎用ライブラリ I/F 呼出し情報）として，以下を一時的に記憶する．

- 汎用ライブラリ I/F を呼出している命令文を特定する情報（ファイル名と行番号）
- 呼出されている汎用ライブラリ I/F の名前

④ 入力データ解析機能において，③で特定した汎用ライブラリを呼出している命令文で，入力データとして汎用ライブラリ I/F に渡される変数について，データフローの解析し，さらにその変数が取りうる値の範囲を解析し一時的に記憶する．なお，リクエスト等の Web アプリケーションの外部からの入力データ（外部入力データ）が格納される変数と，その他のデータは区別して記憶する．

⑤ データ／インタプリタ構文解析機能において，汎用ライブラリ I/F に渡されるデータ（HTML や SQL 文等）の構文上のどの位置に外部入力データが利用されているかを調べ，一時的に記憶する．例えば，HTML 文の場合には，外部入力データが利用される箇所が，どの要素のコンテンツなのか，どの属性の属性値なのか等を調べる（これにより詳細な脆弱性対策の方法が異なる）．

⑥ 最後に動作設定ファイル出力機能において，③～⑤で解析した結果を用いて，対策選択機能の動作設定ファイルに汎用ライブラリを呼出す可能性がある各命令文に関する情報（汎用ライブラリ I/F 呼出し情報）として，対策選択機能の動作設定ファイルを出力する．

- 呼出されている汎用ライブラリ I/F の名前
- 汎用ライブラリ I/F を呼出している命令文を特定する情報（ファイル名と行番号）
- 汎用ライブラリ I/F に渡される引数について，外

部入力データが含まれる箇所の情報，データを無害化するために実行すべきセキュアライブラリの名前

(2) 実行環境での処理フロー

実行環境での処理フローを図 4 に示す．

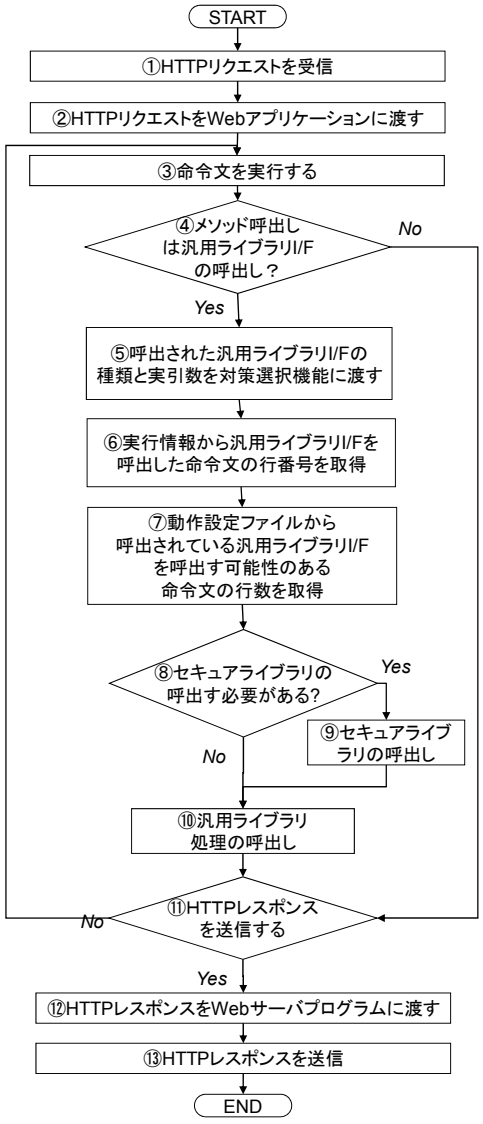

図 4　実行環境での処理フロー

① Web サーバプログラムは，HTTP リクエストを受信する．

② Web サーバプログラムは，受信した HTTP リクエストを Web アプリケーションに渡す．

③ Web アプリケーションでは，受信した HTTP リクエストに基づいて各命令文を実行する．

④ 呼出された命令文が汎用ライブラリ I/F の場合には，⑤を行う．それ以外の場合には，⑪を行う．

⑤ 呼出された汎用ライブラリ I/F は，汎用ライブラリ I/F の名前と，汎用ライブラリ I/F 呼出しの実引数を対策選択機能に渡す．

⑥ 対策選択機能は，汎用ライブラリ I/F を呼出すまでに実行された命令文の実行情報（Java のスタックトレース等）を取得し，汎用ライブラリ I/F の名前を用いて，汎用ライブラリ I/F を呼出した命令文の行番号を取得する．

⑦ 対策選択機能では，動作設定ファイルから⑤で渡された汎用ライブラリ I/F の名前の情報を含む汎用ライブラリ I/F 呼出し情報を取得する．

⑧ 対策選択機能は，セキュアライブラリを呼出す必要があるかを判定する．セキュアライブラリを呼ぶ必要があると判定した場合には⑨を行う．セキュアライブラリを呼ぶ必要がないと判定した場合には⑩を行う．

なお，セキュアライブラリを呼ぶ必要があるか否かの具体的な判定方法は下記の通りである．

1) ⑦で取得した汎用ライブラリ I/F 呼出し情報の中から，⑥で取得した汎用ライブラリ I/F を呼出している命令文の行番号と一致する行番号を含む情報を選択する．
2) 1)で選択した汎用ライブラリ I/F 呼出し情報に含まれる汎用ライブラリ I/F に渡される引数の情報を調べて，外部入力データを無害化するために実行すべきセキュアライブラリの名前を取得する．
3) ⑥で取得した実行情報を調べ，2)で取得したセキュアライブラリが，引数で利用する外部入力データに対して利用されていないかを調べ，実行されていない場合にはセキュアライブラリを実行する必要があると判定する．

⑨ セキュアライブラリでは，汎用ライブラリ I/F に渡された実引数のうち外部入力データが利用される位置と及び実行すべき対策選択機能の情報を使って，外部入力データが利用される位置に含まれる箇所のデータについて，動作設定ファイルで指定された脆弱性対策機能を呼出して外部入力データの無害化等の対策処理を行う．

⑩ 汎用ライブラリ処理機能を呼出す．

⑪ 呼出された命令文の関数名から HTTP レスポンスを送信する処理かを否かを判定する．HTTP レスポンスを送信する場合には⑫を行う．HTTP レスポンスを送信しない場合には，③に戻り次の命令文を実行する．

⑫ Web アプリケーションは HTTP レスポンスを Web サーバプログラムに渡す．

⑬ Web サーバプログラムは，クライアントに HTTP レスポンスを送信する．

Vol.2013-DPS-154 No.27
Vol.2013-CSEC-60 No.27
2013/3/14

4. 考察

本提案方式の適用が Web アプリケーションの脆弱性対策に与える効果と，本提案方式の適用により Web アプリケーションに悪影響を及ぼさないかを考察する．

本提案方法では，運用開始前に Web アプリケーションのソースコードを静的に解析することで得られた情報と Web アプリケーションの実行時に呼ばれた関数等の情報を含む実行情報に基づき，セキュアライブラリでの脆弱性対策の実施を動的に行う．運用開始前に行う静的解析では，汎用ライブラリに渡す HTML 文や SQL 文等の構文上のどの位置に外部入力データが利用されるかを解析することで，汎用ライブラリに渡すデータのどの部分にどのようなセキュアライブラリを適用する必要があるかを調べる．また，Web アプリケーションの実行時には，静的解析で調べた情報に従って，セキュアライブラリでの脆弱性対策を施す．これにより，開発者がプログラムの内容を確認して，セキュアライブラリによる脆弱性対策の処理を実装する必要がなくなるため，開発者のミス等で不適切な脆弱性対策が施されることを防ぐことができると考えられる．

また，セキュアライブラリによる脆弱性対策を行うか否かを判定する際に，事前に同じ対策が施されていないかを対策選択機能が確認するため，過剰に対策処理を施すことによる悪影響を防止する効果も期待できる．

一方で，本提案方式では，Web アプリケーションの実行時に，実行情報を取得し，静的解析で取得した情報との比較する．このため，Web アプリケーションのレスポンスが遅くなる等，実行速度に影響することが懸念される．今後は，どの程度の速度性能に影響がでるかの検証が必要である．

5. おわりに

本稿では，Web アプリケーションの脆弱性対策を正確に行うことを目的に，運用開始前に Web アプリケーションのソースコードを静的解析して得られた情報を用いて，脆弱性対策を行うセキュアライブラリの選択を動的に行う方法を提案した．

本提案方式では，静的解析で得られた情報に基づいて脆弱性対策を行うセキュアライブラリを選択するため，開発者のミス等により不適切な脆弱性対策を施すことを防止できると考えられる．また，静的解析で得た情報と，Web アプリケーションの実行時に取得した情報に基づき，脆弱性対策を行うセキュアライブラリの呼出しの要否を決定するため，脆弱性対策の重複した実行による悪影響を防ぐ効果も期待できる．

今後は，本提案方式の適用による Web アプリケーションの実行速度への影響を検証する予定である．

参考文献

1) 独立行政法人 情報処理推進機構：組織内部者の不正行為によるインシデント調査 -調査報告書- ，
http://www.ipa.go.jp/security/fy23/reports/insider/documents/insider_report.pdf
2) 独立行政法人 情報処理推進機構：ソフトウエア等の脆弱性情報に関する届出状況［2012 年第 3 四半期（7 月～9 月）］，
http://www.ipa.go.jp/security/vuln/report/vuln2012q3.html
3) The Open Web Application Security Project：OWASP Top Ten Project,
https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
http://office.microsoft.com/ja-jp/products
4) The Open Web Application Security Project：OWASP Code Review Project,
https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
5) The Open Web Application Security Project：PHP Top 5,
https://www.owasp.org/index.php/PHP_Top_5
6) 独立行政法人 情報処理推進機構：セキュア・プログラミング講座：Web アプリケーション編,
http://www.ipa.go.jp/security/awareness/vendor/programmingv2/index.html
7) 独立行政法人 情報処理推進機構：安全なウェブサイトの作り方，http://www.ipa.go.jp/security/vuln/documents/website_security.pdf.
8) 独立行政法人 情報処理推進機構：安全な SQL の呼び出し方，
http://www.ipa.go.jp/security/vuln/documents/website_security_sql.pdf
9) Web Application Security Consortium（WASC）：The WASC Threat Classification v2.0,
http://projects.webappsec.org/w/page/13246978/Threat%20Classification
10) Common Weakness Enumeration：CWE/SANS Top 25 Most Dangerous Software Errors，http://cwe.mitre.org/top25/
11) OWASP Enterprise Security API：
https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API
12) Microsoft Anti-Cross Site Scripting Library V4.2:
http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=28589
13) A.S.Christensen,A.Moller,and M.I.Schwartzbach：Precise Analysis of String Expressions， In SAS 2003
14) Y.Minamide：Static Approximation of Dynamically Generated Web Pages， In WWW 2005
15) G.Wassermann and Z. Su.：Sound and Precise Analysis of Web Applications for Injection Vulnerabilities， In PLDI 2007
16) V.B.Livshits and M.S.Lam：Finding Security Vulnerabilities in Java Applications with Static Analysis”，In USENIX Security 2005