(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5863973号
(P5863973)
(45)発行日　平成28年2月17日(2016.2.17)　(24)登録日　平成28年1月8日(2016.1.8)

(51)Int.Cl.　F I
G06F 21/12 (2013.01)　G06F 21/12 380

請求項の数 7 (全 19 頁)

(21)出願番号　特願2014-528105(P2014-528105)
(86)(22)出願日　平成25年7月25日(2013.7.25)
(86)国際出願番号　PCT/JP2013/070188
(87)国際公開番号　WO2014/021190
(87)国際公開日　平成26年2月6日(2014.2.6)
審査請求日　平成26年8月8日(2014.8.8)
(31)優先権主張番号　特願2012-171164(P2012-171164)
(32)優先日　平成24年8月1日(2012.8.1)
(33)優先権主張国　日本国(JP)

(73)特許権者　000006013
三菱電機株式会社
東京都千代田区丸の内二丁目７番３号
(74)代理人　100099461
弁理士　溝井　章司
(74)代理人　100152881
弁理士　山地　博人
(72)発明者　植田　武
日本国東京都千代田区丸の内二丁目７番３号　三菱電機株式会社内
(72)発明者　桜井　鐘治
日本国東京都千代田区丸の内二丁目７番３号　三菱電機株式会社内

審査官　宮司　卓佳

最終頁に続く

(54)【発明の名称】プログラム実行装置及びプログラム解析装置

(57)【特許請求の範囲】
【請求項１】
ライブラリ関数を用いるプログラムを実行するプログラム実行装置であって、
前記プログラムで用いられるライブラリ関数のうち脆弱性のあるライブラリ関数である脆弱性ライブラリ関数の脆弱性に対する対策処理を行う脆弱性対策処理部と、
前記プログラム実行時に前記脆弱性ライブラリ関数の呼び出しが要求された場合に、前記脆弱性ライブラリ関数の脆弱性に対する対策処理を前記脆弱性対策処理部に行わせ、前記脆弱性対策処理部により対策処理が行われた後に、前記脆弱性ライブラリ関数を呼び出す脆弱性対策制御部とを有することを特徴とするプログラム実行装置。
【請求項２】
前記プログラム実行装置は、更に、
前記脆弱性ライブラリ関数で処理されるパラメータ値における不正値が示される脆弱性情報を記憶する脆弱性情報記憶部を有し、
前記脆弱性対策制御部は、
前記脆弱性ライブラリ関数の呼び出しが要求された場合に、当該呼び出しによって前記脆弱性ライブラリ関数で処理されるパラメータ値が前記不正値に該当するか否かを判断し、
前記脆弱性ライブラリ関数で処理されるパラメータ値が前記不正値に該当する場合は、前記脆弱性ライブラリ関数が前記不正値を処理しないようにする対策処理を前記脆弱性対策処理部に行わせることを特徴とする請求項１に記載のプログラム実行装置。

【請求項３】
前記脆弱性対策制御部は、
前記脆弱性ライブラリ関数以外のライブラリ関数の呼び出しが要求された場合は、前記脆弱性対策処理部に対策処理を行わせることなく、対象のライブラリ関数を呼び出し、
前記脆弱性ライブラリ関数の呼び出しが要求され、当該呼び出しによって前記脆弱性ライブラリ関数で処理されるパラメータ値が前記不正値に該当しない場合は、前記脆弱性対策処理部に対策処理を行わせることなく、前記脆弱性ライブラリ関数を呼び出すことを特徴とする請求項２に記載のプログラム実行装置。
【請求項４】
前記プログラム実行装置は、
複数の脆弱性ライブラリ関数に対応させて複数の脆弱性対策処理部を有し、
前記脆弱性対策制御部は、
前記プログラム実行時に、いずれかの脆弱性ライブラリ関数の呼び出しが要求された場合に、呼び出しの対象の脆弱性ライブラリ関数に対応する脆弱性対策処理部を選択し、
選択した脆弱性対策処理部に、脆弱性に対する対策処理を行わせることを特徴とする請求項１～３のいずれかに記載のプログラム実行装置。
【請求項５】
前記プログラム実行装置は、
外部装置から受信した受信データに記述されているパラメータ値を前記脆弱性ライブラリ関数に処理させ、
前記プログラム実行装置は、更に、
前記脆弱性ライブラリ関数で処理されるパラメータ値における不正値が示されるとともに、前記脆弱性ライブラリ関数で処理されるパラメータ値の受信データにおける記述位置ごとに、対策処理が示される脆弱性情報を記憶する脆弱性情報記憶部を有し、
前記脆弱性対策制御部は、
外部装置からの受信データの受信に伴って前記脆弱性ライブラリ関数の呼び出しが要求され、当該呼び出しによって前記脆弱性ライブラリ関数で処理されるパラメータ値が前記不正値に該当する場合に、
当該パラメータ値の受信データにおける記述位置を判定し、当該パラメータ値の記述位置に対応する対策処理を前記脆弱性対策処理部に行わせることを特徴とする請求項１～４のいずれかに記載のプログラム実行装置。
【請求項６】
前記脆弱性対策処理部と前記脆弱性対策制御部が、汎用ライブラリに含まれていることを特徴とする請求項１～５のいずれかに記載のプログラム実行装置。
【請求項７】
プログラム及び前記プログラムで用いられるライブラリ関数を解析して、外部入力データに対して脆弱性のあるライブラリ関数を脆弱性ライブラリ関数として抽出するとともに、
前記プログラムの設定ファイルの値の適否を検査するための試験データ及びスクリプトの少なくともいずれかを、前記設定ファイルの値と、前記脆弱性ライブラリ関数と、外部入力データが利用される前記プログラム内のコードの位置とを用いて生成することを特徴とするプログラム解析装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、プログラムの脆弱性に対する対策処理を行う技術に関する。
【背景技術】
【０００２】
従来から、ＯＳ（Ｏｐｅｒａｔｉｎｇ　Ｓｙｓｔｅｍ）やアプリケーションプログラム（以下、単に「アプリケーション」という）など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などの被害が発生している。
近年では、Ｗｅｂサーバ上で動作するＷｅｂアプリケーションへの攻撃が増加しており、機密情報や個人情報の漏えいや改ざん、サービスの不正利用などが社会的な問題となっている。
Ｗｅｂアプリケーションに対する攻撃は、Ｗｅｂアプリケーションに対するクライアントからの入力データとなるＨＴＴＰ（ＨｙｐｅｒＴｅｘｔ Ｔｒａｎｓｆｅｒ Ｐｒｏｔｏｃｏｌ）のリクエスト中に、不正なデータを混入させることで行う。
Ｗｅｂアプリケーションの脆弱性は、入力データを利用する処理内容に応じて異なり、クロスサイトスクリプティングやＳＱＬインジェクションなどの複数の種類があることが知られている。

【０００３】
例えば、クロスサイトスクリプティングは、ＨＴＭＬ（ＨｙｐｅｒＴｅｘｔ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ）データを動的に生成する処理における脆弱性を利用した攻撃であり、この脆弱性により、ＨＴＭＬデータ中にスクリプトが混入することを許してしまう。
また、ＳＱＬインジェクションは、リレーショナルデータベースのデータ操作を行うＳＱＬ文を発行する処理の脆弱性を利用した攻撃であり、この脆弱性により、アプリケーションが想定しないＳＱＬ文が実行されることを許してしまう。

【０００４】
脆弱性を攻撃するための不正なデータは、脆弱性の種類によって異なり、それらの対策方法も異なる。
例えば、クロスサイトスクリプティングを行うための方法の１つとして、不正なデータとして＜ＳＣＲＩＰＴ＞タグをリクエスト中に含める方法がある。
クロスサイトスクリプティングの対策としては、生成するＨＴＭＬ中で＜ＳＣＲＩＰＴ＞タグが＜ＳＣＲＩＰＴ＞タグとして解釈されないようにＨＴＭＬ文法に則り無害化することである。
また、ＳＱＬインジェクションでは、文字列を括るための「’」を混入することで文字列を終端し、その後に任意のＳＱＬ文を混入する方法がある。
このようなＳＱＬインジェクションに対する対策は、入力データに含まれる「’」が、文字列を括る記号であることを意味しないようにＳＱＬの文法に則り無害化することである。

【０００５】
従来では、Ｗｅｂアプリケーションの脆弱性に対する攻撃を防御するためには、主に以下の２つの対策が採られている。
１つは、各種脆弱性を発生させる入力データを検証して、不正なデータを無害化するセキュリティ機能を、脆弱性が発生しうる処理内容に応じてＷｅｂアプリケーション自体に組み込むことである。
２つ目の防御方法は、ＷｅｂアプリケーションへのＨＴＴＰリクエストをＷｅｂアプリケーションに到達する前に検査して、攻撃が発生する可能性があるＨＴＴＰリクエストのブロックや無害化を行うＷｅｂアプリケーションファイアウォールを利用することである。

【０００６】
１つ目の防御方法については、ＯＷＡＳＰ（Ｔｈｅ Ｏｐｅｎ Ｗｅｂ Ａｐｐｌｉｃａｔｉｏｎ Ｓｅｃｕｒｉｔｙ Ｐｒｏｊｅｃｔ）などの組織が組み込むセキュリティ機能などのガイドラインを公開している。
また、セキュリティ機能の組み込みを支援する方法として、プログラムのソースコードを解析することで、脆弱性が発生しうる箇所を検出する方法も知られている（例えば、特許文献１）。

【０００７】

２つ目の防御方法のＷｅｂアプリケーションファイアウォールについては、事前に登録してある不正なデータのパターンが含まれているかを検証し、不正なデータのパターンがＨＴＴＰリクエストに含まれていた場合には、通信の遮断やデータの無害化などを行う方法が一般的である（例えば、非特許文献１）。

また、各種の脆弱性対策を行うライブラリを用いる方法がある（例えば、特許文献２）。

この方法では、開発者が作成したリクエストの各パラメータに対する各種脆弱性の対策の要否を記載した設定ファイルを読込み、設定ファイルの内容に応じてＨＴＴＰリクエストの受信時にパラメータごとに指定された脆弱性対策を行うライブラリを利用することで、Ｗｅｂアプリケーションの脆弱性対策を効率的に行うことができる。

【先行技術文献】

【特許文献】

【０００８】

【特許文献１】特開２００７－５２６２５号公報

【特許文献２】特開２００７－４７８８４号公報

【非特許文献】

【０００９】

【非特許文献１】ＭｏｄＳｅｃｕｒｉｔｙ　＜ＵＲＬ：ｈｔｔｐ：／／ｗｗｗ．ｍｏｄｓｅｃｕｒｉｔｙ．ｏｒｇ／＞

【発明の概要】

【発明が解決しようとする課題】

【００１０】

従来のＷｅｂアプリケーション内のソースコードを静的に解析する方法（特許文献１）は、脆弱性が発生しうる箇所は検出するものの、脆弱性に対する対策方法の検討と修正は、Ｗｅｂアプリケーションの開発者が個別に行わなければいけないという課題がある。

【００１１】

また、Ｗｅｂアプリケーションファイアウォールによって対策する方法（非特許文献１）は、Ｗｅｂアプリケーションが行う処理内容を考慮しない。

従って、Ｗｅｂアプリケーションの処理内容によっては特定の種類の脆弱性が問題にならない場合やＷｅｂアプリケーションにおいて脆弱性の対策を施している場合も、不正なデータのパターンに該当するものの無害なデータが不正なデータとして誤検出されてしまう。

このため、Ｗｅｂアプリケーションでは実質問題がない通信を遮断することや、無害化によって、正常に扱えるデータが変換されてしまう悪影響が発生するという課題がある。

【００１２】

さらに、リクエストのパラメータごとに対策する方法（特許文献２）は、設定ファイルを開発者がパラメータごとに設定する必要があり手間がかかるという課題がある。

さらに、設定ミスなどにより、脆弱性の対策抜けや不必要な対策を選択してしまう可能性がある。

さらに、指定された脆弱性対策は、Ｗｅｂアプリケーションファイアウォールと同様にＨＴＴＰリクエスト受信時に一括して行われるため、施された脆弱性対策が別の脆弱性の可能性がある処理に悪影響を及ぼすという課題もある。

【００１３】

この発明は上記のような課題を解決することを主な目的としており、脆弱性を狙った攻撃がなされる前に脆弱性に対する対策処理を確実に実行し、対策漏れを防止し、不要な対策処理による悪影響を回避できる構成を得ることを主な目的とする。

【課題を解決するための手段】

【００１４】

本発明に係るプログラム実行装置は、

ライブラリ関数を用いるプログラムを実行するプログラム実行装置であって、

前記プログラムで用いられるライブラリ関数のうち脆弱性のあるライブラリ関数である脆弱性ライブラリ関数の脆弱性に対する対策処理を行う脆弱性対策処理部と、
前記プログラム実行時に前記脆弱性ライブラリ関数の呼び出しが要求された場合に、前記脆弱性ライブラリ関数の脆弱性に対する対策処理を前記脆弱性対策処理部に行わせ、前記脆弱性対策処理部により対策処理が行われた後に、前記脆弱性ライブラリ関数を呼び出す脆弱性対策制御部とを有することを特徴とする。

【発明の効果】

【００１５】

本発明によれば、プログラム実行時に脆弱性ライブラリ関数の呼び出しが要求された場合に、当該脆弱性ライブラリ関数の脆弱性に対する対策処理を行うため、脆弱性を狙った攻撃がなされる前に脆弱性に対する対策処理を確実に実行することができ、対策漏れを防止し、不要な対策処理による悪影響を回避することができる。

【図面の簡単な説明】

【００１６】

【図１】実施の形態１に係るシステム構成例を示す図。
【図２】実施の形態１に係るＷｅｂアプリケーション脆弱性対策システムの構成例を示す図。
【図３】実施の形態１に係る開発環境における動作例を示すフローチャート図。
【図４】実施の形態１に係る実行環境における動作例を示すフローチャート図。
【図５】実施の形態２に係るＷｅｂアプリケーション脆弱性対策システムの構成例を示す図。
【図６】実施の形態２に係る開発環境における動作例を示すフローチャート図。
【図７】実施の形態３に係るＷｅｂアプリケーション脆弱性対策システムの構成例を示す図。
【図８】実施の形態３に係る開発環境における動作例を示すフローチャート図。
【図９】実施の形態１～３に係るＷｅｂアプリケーション脆弱性対策システムのハードウェア構成例を示す図。

【発明を実施するための形態】

【００１７】

実施の形態１．

本実施の形態及び以降の実施の形態では、必要十分なＷｅｂアプリケーションのセキュリティ対策を自動選択し、対策漏れの防止と脆弱性対策による悪影響を回避するＷｅｂアプリケーション脆弱性対策システムを説明する。

【００１８】

図１は、本実施の形態に係るＷｅｂアプリケーション脆弱性対策システムを含むシステム構成図である。

図１に示すように、本実施の形態に係るシステムは、Ｗｅｂアプリケーション脆弱性対策システム１０００、クライアント端末２０００、ネットワーク３０００（インターネット等のネットワーク）から構成される。

Ｗｅｂアプリケーション脆弱性対策システム１０００と複数のクライアント端末２０００は、ネットワーク３０００を介して接続される。

【００１９】

また、図２は、本実施の形態のＷｅｂアプリケーション脆弱性対策システム１０００の構成図である。

なお、図２では、図示を省略しているが、Ｗｅｂアプリケーション脆弱性対策システム１０００は、ＣＰＵ（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）、メモリ、二次記憶装置等のハードウェア資源を有している。

そして、以下に説明する機能を実現するプログラムを二次記憶装置に格納しておき、プログラムの実行に際して、二次記憶装置からプログラムをメモリにロードし、ＣＰＵがメモリにロードされたプログラムを実行する。

【００２０】
図２において、Ｗｅｂアプリケーション脆弱性対策システム１０００は、開発環境１１００と実行環境１２００から構成される。
開発環境１１００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１２００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。
実行環境１２００は、プログラム実行装置の例に相当する。
【００２１】
開発環境１１００は、静的コード解析部１１２０を備える。
静的コード解析部１１２０は、Ｗｅｂアプリケーションのソースコード又は実行コード（「Ｗｅｂアプリケーションのソース／実行コード１１０１」と表記する）を読込み、Ｗｅｂアプリケーションのソース／実行コード１１０１を静的解析して、脆弱性対策のための動作設定ファイル１１３０を出力する。
また、静的コード解析部１１２０は、Ｗｅｂアプリケーション入力部１１２１、汎用ライブラリ利用検証部１１２２、外部入力データ解析部１１２３、動作設定ファイル出力部１１２４から構成される。
【００２２】
Ｗｅｂアプリケーション入力部１１２１は、Ｗｅｂアプリケーションのソース／実行コード１１０１を読込む。
汎用ライブラリ利用検証部１１２２は、Ｗｅｂアプリケーションのソース／実行コード１１０１に対する静的コード解析により汎用ライブラリ関数を呼び出しているコードの位置と汎用ライブラリＩ／Ｆ１２３１の種類を調べて、汎用ライブラリ関数の利用方法が正しいかを検証する。
外部入力データ解析部１１２３は、静的コード解析により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）のデータフローを解析し、その外部入力データの利用されるコードの位置と取り得る値（データの値及びデータの型）を解析する。
動作設定ファイル出力部１１２４は、汎用ライブラリ利用検証部１１２２と外部入力データ解析部１１２３の解析結果に基づいて、汎用ライブラリの実行時の動作を決定するための動作設定ファイル１１３０を出力する。
なお、動作設定ファイル１１３０には、脆弱性のある汎用ライブラリ関数（以下、「脆弱性ライブラリ関数」という）と、脆弱性ライブラリ関数で処理された場合に異常が発生し得るパラメータの名称と当該パラメータにおける不正値と、脆弱性ライブラリ関数を呼び出すコード位置のうち異常が発生し得るコード位置とが記述されている。
動作設定ファイル１１３０は、脆弱性情報の例に相当する。
【００２３】
また、実行環境１２００は、Ｗｅｂサーバプログラム１２１０、Ｗｅｂアプリケーション１２２０、汎用ライブラリ１２３０、動作設定ファイル記憶部１２４０から構成される。
【００２４】
Ｗｅｂサーバプログラム１２１０は、Ｗｅｂアプリケーション１２２０を実行するためのプログラムである。
【００２５】
Ｗｅｂアプリケーション１２２０は、汎用ライブラリ関数を呼び出し、汎用ライブラリ関数の実行結果を得る。
【００２６】
汎用ライブラリ１２３０は、ＨＴＴＰリクエストの送受信や、ＨＴＭＬ出力処理や、データベースへのアクセス、ファイルの読み書きなどの一般的なＷｅｂアプリケーションで行われる処理を行うための機能を提供する。
【００２７】

動作設定ファイル記憶部１２４０は、開発環境１１００の静的コード解析部１１２０より出力された動作設定ファイル１１３０を記憶する。
動作設定ファイル記憶部１２４０は、脆弱性情報記憶部の例に相当する。
【００２８】
汎用ライブラリ１２３０は、汎用ライブラリＩ／Ｆ（インタフェース）１２３１、対策選択部１２３２、脆弱性対策処理部１２３３、汎用ライブラリ処理部１２３４から構成される。
【００２９】
汎用ライブラリＩ／Ｆ１２３１は、Ｗｅｂアプリケーション１２２０が汎用ライブラリ１２３０を利用するためのプログラムインタフェースを提供する。
なお、汎用ライブラリＩ／Ｆ１２３１は、ＨＴＭＬ出力処理やデータベースアクセスなどの処理の分類ごとに用意される。
【００３０】
脆弱性対策処理部１２３３は、Ｗｅｂアプリケーションの脆弱性に対する対策処理を行う。
なお、脆弱性対策処理部１２３３は、脆弱性の対策ごとに複数用意される。
つまり、脆弱性対策処理部１２３３は、ＨＴＭＬ出力処理やデータベースアクセスなどの処理の分類ごとに用意される。
【００３１】
汎用ライブラリ処理部１２３４は、汎用ライブラリ１２３０が提供する処理（ＨＴＭＬ出力処理やデータベースアクセスなど）を行う。
汎用ライブラリ処理部１２３４は、汎用ライブラリ関数に相当する。
また、汎用ライブラリ処理部１２３４は、処理内容に応じて複数用意される。
【００３２】
対策選択部１２３２は、動作設定ファイル１１３０を読み込み、汎用ライブラリ１２３０内部で脆弱性の対策処理が必要か否かを判定する。
より具体的には、対策選択部１２３２は、Ｗｅｂアプリケーション１２２０から汎用ライブラリ関数の呼び出しが要求された場合、すなわち、汎用ライブラリ処理部１２３４の呼び出しが要求された場合に、動作設定ファイル１１３０の内容に基づき、脆弱性の対策処理が必要か否かを判定する。
そして、脆弱性の対策処理が必要な場合は、対策選択部１２３２は、まず、脆弱性対策処理部１２３３に脆弱性の対策処理を行わせ、脆弱性対策処理部１２３３により対策処理が行われた後に、対象の汎用ライブラリ処理部１２３４を呼び出す。
対策選択部１２３２は、脆弱性対策制御部の例に相当する。
【００３３】
次に、Ｗｅｂアプリケーション脆弱性対策システム１０００で行う動作を説明する。
Ｗｅｂアプリケーション脆弱性対策システム１０００では、最初に開発環境１１００の静的コード解析部１１２０でＷｅｂアプリケーションのソース／実行コード１１１０を静的コード解析し、動作設定ファイル１１３０を生成する。
そして、Ｗｅｂアプリケーションのソース／実行コード１１１０をコンパイルして実行可能にしてＷｅｂアプリケーション１２２０を実行環境１２００に配置し、また、動作設定ファイル１１３０を実行環境１２００に配置し、Ｗｅｂアプリケーション１２２０の運用が開始される。
【００３４】
開発環境１１００での動作例を図３を用いて説明する。
【００３５】
開発環境１１００では、Ｗｅｂアプリケーション入力部１１２１でＷｅｂアプリケーションのソース／実行コード１１１０が読込まれる（Ｓ１０１）。
次に、汎用ライブラリ利用検証部１１２２が、Ｗｅｂアプリケーションのソース／実行コード１１１０が利用する汎用ライブラリ関数を呼び出しているコードの位置（ライン数

など）、呼び出しされる汎用ライブラリ関数の種類（ＨＴＭＬ出力、データベースアクセスなど）を調べた上で、汎用ライブラリ関数に与えるデータや使い方が適切であるかを検証する（Ｓ１０２）。

汎用ライブラリ関数が適切に利用されていると判定した場合には（Ｓ１０２のＹｅｓ）、外部入力データ解析部１１２３が、外部からの入力データが格納される変数（外部入力データ）を特定し、外部入力データのデータフローを調べて、外部入力データが利用される一連のコードの位置及び、各コードにおける外部入力データが取り得る値の範囲を調べる（Ｓ１０３）。

汎用ライブラリ関数が適切に利用されていないと判定した場合には（Ｓ１０２のＮｏ）、適切に利用されていない汎用ライブラリ関数の種類、コード上の位置、適切に利用されていないと判定した理由などを含むメッセージを出力して処理を終了する（Ｓ１０４）。

【００３６】

Ｓ１０３の後には、動作設定ファイル出力部１１２４が、Ｓ１０２で解析した汎用ライブラリ関数を呼び出しているコードの位置及び呼び出される汎用ライブラリ関数の種類並びに、Ｓ１０３で解析した外部入力データが利用される一連のコードの位置及び各コードにおける外部入力データが取り得る値の結果に基づいて、動作設定ファイル１１３０を出力する（Ｓ１０５）。

この動作設定ファイル１１３０は、外部入力データの取り得る値によって脆弱性が発生する可能性がある汎用ライブラリ関数を呼び出すコードの位置を含む、対策選択のためのファイルである。

Ｓ１０５の処理をさらに具体的に説明すると、Ｓ１０２で解析した汎用ライブラリ関数を呼び出しているコード位置のうちＳ１０３で解析した外部入力データが利用される一連のコード位置に含まれているコード位置を抽出し、そのコード位置のコードが呼び出している汎用ライブラリ関数に、Ｓ１０３で解析した外部入力データの取り得る値の範囲を入力した場合に脆弱性が問題になるか否か（異常が発生するか否か）を外部入力データの値ごとに判定する。

そして、脆弱性が問題になると判定した場合には、動作設定ファイル出力部１１２４は、脆弱性のある汎用ライブラリ関数（脆弱性ライブラリ関数）の名称と、脆弱性ライブラリ関数を呼び出すコード位置と、外部入力データの種類（変数の名称）と、外部入力データの不正値（異常が発生する値）が記述される動作設定ファイル１１３０を生成する。

そして、生成された動作設定ファイル１１３０は、動作設定ファイル記憶部１２４０に出力され、動作設定ファイル記憶部１２４０に格納される。

【００３７】

次に、実行環境１２００での動作例を、図４を用いて説明する。

【００３８】

まず、Ｗｅｂサーバプログラム１２１０は、クライアント端末２０００からのＨＴＴＰリクエストを受信する（Ｓ２０１）。

Ｗｅｂサーバプログラム１２１０は、受信したＨＴＴＰリクエストをＷｅｂアプリケーション１２２０に渡す（Ｓ２０２）。

Ｗｅｂアプリケーション１２２０が、汎用ライブラリＩ／Ｆ１２３１を呼び出し、汎用ライブラリ関数の呼び出しを要求した場合に（Ｓ２０３のＹｅｓ）、呼び出された汎用ライブラリＩ／Ｆ１２３１は対策選択部１２３２を呼び出す（Ｓ２０４）。

対策選択部１２３２は、動作設定ファイル１１３０を読込み、動作設定ファイル１１３０から、脆弱性を顕在化させる外部入力データが混入するおそれがある汎用ライブラリ関数（脆弱性ライブラリ関数）の呼び出しコード位置の情報を取得する（Ｓ２０５）。

次に、対策選択部１２３２は、Ｓ２０３で汎用ライブラリＩ／Ｆ１２３１を呼び出すまでに実行されたコードの位置の情報をスタックトレースなどの実行情報から取得する（Ｓ２０６）。

【００３９】

次に、対策選択部１２３２は、脆弱性対策処理部１２３３を実行する必要があるか否か

を判定する（Ｓ２０７）。
具体的には、Ｓ２０５で取得したコード位置の中にＳ２０６で取得したコード位置と一致するものがあれば、動作設定ファイル１１３０から、Ｓ２０６で取得したコード位置と一致するコード位置に対応付けられている脆弱性ライブラリ関数の名称と、外部入力データの種類と、外部入力データの不正値を取得する。
そして、対策選択部１２３２は、取得した脆弱性ライブラリ関数の名称と外部入力データの種類と外部入力データの不正値と、Ｗｅｂアプリケーション１２２０が呼び出そうとしている汎用ライブラリ関数の名称と当該汎用ライブラリ関数で処理させようとしている外部入力データの種類と外部入力データの値とを照合し、３つの要素のすべてが一致する場合に、脆弱性対策処理部１２３３を実行する必要があると判定する。
一方、３つの要素の少なくとも１つが一致しない場合は、脆弱性対策処理部１２３３を実行する必要がないと判定する。
また、対策選択部１２３２は、Ｗｅｂアプリケーション１２２０が呼び出そうとしている脆弱性ライブラリ関数に対応する脆弱性対策処理部１２３３を呼び出しの対象として指定する。
【００４０】
対策選択部１２３２は、脆弱性対策処理部１２３３を呼び出す必要があると判定した場合（Ｓ２０７のＹＥＳ）には、汎用ライブラリ処理部１２３４の呼び出しを行う前に、脆弱性対策処理部１２３３の呼び出しを行い、汎用ライブラリ処理部１２３４が外部入力データの不正値を処理しないようにする対策処理を脆弱性対策処理部１２３３に行わせる（Ｓ２０８）。
具体的には、脆弱性対策処理部１２３３に外部入力データの無害化などの対策処理を行わせる。
脆弱性対策処理部１２３３による対策処理が完了した後に、対策選択部１２３２は、汎用ライブラリ処理部１２３４を呼び出す（Ｓ２０９）。
【００４１】
脆弱性対策処理部１２３３の呼び出しが必要ないと判断した場合（Ｓ２０７のＮＯ）には、対策選択部１２３２は、脆弱性対策処理部１２３３の呼び出しは行わずに、汎用ライブラリ処理部１２３４の呼び出しを行う（Ｓ２０９）。
【００４２】
以上のＳ２０３～Ｓ２０９のステップは、Ｗｅｂサーバプログラム１２１０がクライアント端末２０００にレスポンスを送信するまでに汎用ライブラリＩ／Ｆ１２３１が呼び出されるたびに行われる（Ｓ２１０）。
そして、Ｗｅｂサーバプログラム１２１０がＨＴＴＰレスポンスをクライアント端末２０００に送信する場合には、Ｗｅｂアプリケーション１２２０はＨＴＴＰレスポンスをＷｅｂサーバプログラム１２１０に渡す（Ｓ２１１）。
そして、最後に、Ｗｅｂサーバプログラム１２１０は、ＨＴＴＰレスポンスをクライアント端末２０００に応答する（Ｓ２１２）。
【００４３】
なお、開発環境１００において、静的コード解析部１１２０がＷｅｂアプリケーションのソースコードを用いて動作設定ファイル１１３０を生成した場合は、動作設定ファイル１１３０にはソースコード上のコード位置（ライン番号等）が記述される。
一方、実行環境１２００では、対策選択部１２３２は、コンパイル後のＷｅｂアプリケーションのコード位置をスタックトレースなどの実行情報から取得する。
このため、上記のＳ２０７では、より詳細には、対策選択部１２３２は、動作設定ファイル１１３０のソースコードでのコード位置をコンパイル後のコード位置に変換し、変換後のコード位置と、スタックトレース等により取得したコンパイル後のコード位置とを照合している。
もしくは、対策選択部１２３２は、スタックトレース等により取得したコンパイル後のコード位置をソースコード上のコード位置に変換し、変換後のコード位置と、動作設定フ

ァイル１１３０のソースコードでのコード位置とを照合している。
このようなコード位置の変換は既存技術により可能である。
【００４４】
以上のように、本実施の形態では、運用開始前にＷｅｂアプリケーションのソースコードもしくは実行コードを静的コード解析し、静的コード解析で得られたデータフローに基づいて、脆弱性対策を自動選択する機能を備えたＷｅｂアプリケーション実行環境を提供する。
これにより、必要十分なＷｅｂアプリケーションの脆弱性対策を自動選択できるため、対策漏れが防止できる上、不要な対策処理を実行しないため、対策処理による悪影響や処理速度の性能劣化を最小限にできる。
【００４５】
本実施の形態では、Ｗｅｂアプリケーションや設定ファイルの静的コード解析の結果を使い、Ｗｅｂアプリケーションの脆弱性対策の必要性の有無を判定し、その判定結果に応じてＷｅｂアプリケーションの脆弱性対策を実施する機能をもつＷｅｂアプリケーション実行環境を有するＷｅｂアプリケーション脆弱性対策システムを説明した。
また、本実施の形態では、静的コード解析で得られた脆弱性対策が必要なライブラリの呼び出しの位置とＷｅｂアプリケーションの実行時に取得した呼び出されたライブラリの位置情報を比較した結果に基づいて、内部で行う動作を変更するライブラリを有することを説明した。
【００４６】
実施の形態２.
以上の実施の形態１では、運用開始前に外部入力データを利用する汎用ライブラリの呼び出し位置を静的コード解析で取得し、Ｗｅｂアプリケーションの実行時に運用開始前に取得した汎用ライブラリの呼び出し位置の情報と、スタックトレースなどのＷｅｂアプリケーションの実行情報に含まれる呼び出されたコードの位置の情報から、汎用ライブラリの内部で行う脆弱性対策の要否を動的に変更する構成を説明した。
次に、本実施の形態では、運用開始前の静的コード解析で、汎用ライブラリに渡されるデータ（ＨＴＭＬやＳＱＬ文など）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ることで、実行時に行う脆弱性対策の正確性を向上する実施の形態を示す。
【００４７】
図５は、本実施の形態における、Ｗｅｂアプリケーション脆弱性対策システム１０００の構成図である。
【００４８】
図５は、実施の形態１の構成（図２）に対して、静的コード解析部１１２０に構文解析部１１２５を追加したものである。
構文解析部１１２５は、外部入力データ解析部１１２３から呼び出されて、ＨＴＭＬ文やＳＱＬ文などのＷｅｂアプリケーションとは異なる言語の構文を解析する機能を提供する。
【００４９】
開発環境１１００での動作を図６を用いて説明する。
【００５０】
図６のＳ３０１～Ｓ３０４は、図３のＳ１０１～Ｓ１０４と同様のフローであるため説明は省略する。
Ｓ３０５では、構文解析部１１２５が、汎用ライブラリ関数に渡されるデータ（ＨＴＭＬやＳＱＬ文など）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などを調べる。
次に、動作設定ファイル出力部１１２４が、Ｓ３０２で解析した汎用ライブラリ関数を呼び出しているコードの位置及び呼び出される汎用ライブラリ関数の名称並びに、Ｓ３０３で解析した外部入力データが利用される一連のコードの位置及び各コードにおける外部

入力データが取り得る値、及びＳ３０５で調べた汎用ライブラリ関数に渡される外部入力データが構文上のどの位置で利用されるかの情報に基づいて、動作設定ファイル１１３０を出力する（Ｓ３０６）。

本実施の形態で生成される動作設定ファイル１１３０には、実施の形態１の記述内容に加えて、脆弱性ライブラリ関数に渡される外部入力データがＨＴＭＬ文又はＳＱＬ文の構文上のどの位置に含まれるかが記述される。

本実施の形態の動作設定ファイル１１３０では、この構文上の位置が複数列挙され、更に、構文上の位置ごとに対策処理が記述されている。

なお、実施の形態１の動作設定ファイル１１３０の記述内容は、脆弱性のある汎用ライブラリ関数（脆弱性ライブラリ関数）の名称と、脆弱性ライブラリ関数を呼び出すコード位置と、外部入力データの種類（変数の名称）と、外部入力データの不正値（異常が発生する値）である。

生成された動作設定ファイル１１３０は、実施の形態１と同様に、動作設定ファイル記憶部１２４０に出力され、動作設定ファイル記憶部１２４０に格納される。

【００５１】

実行環境１２００の動作では、図４の処理フローのＳ２０８の処理が、実施の形態１と異なる。

【００５２】

脆弱性対策処理部１２３３を呼び出す必要があると判定した場合（Ｓ２０７のＹＥＳ）に、対策選択部１２３２は、Ｓ２０８において、無害化の対象となる外部入力データがクライアント端末２０００からのＨＴＴＰリクエスト内のＨＴＭＬ文又はＳＱＬ文のどの位置に記述されているかを判断する。

更に、対策選択部１２３２は、動作設定ファイル１１３０に記述されている複数の構文位置の中から、ＨＴＴＰリクエストにおける記述位置と合致するものを抽出し、抽出した構文位置の対策処理を抽出する。

そして、対策選択部１２３２は、脆弱性対策処理部１２３３に、抽出した対策処理を実行させて、外部入力データの無害化等を行わせる。

これ以外のフローは、図４と同じ処理フローであるため説明は省略する。

【００５３】

以上のように、本実施の形態では、静的コード解析において、汎用ライブラリに渡すＨＴＭＬ文やＳＱＬ文など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調べることで、汎用ライブラリの内部で実行する脆弱性対策の内容をより正確に選択できるようにできる。

【００５４】

以上、本実施の形態では、静的コード解析で得られた外部からの入力データがライブラリに渡されるデータの構文上のどこに利用されるかの情報に基づいて、内部で行う動作を変更するライブラリを有することを説明した。

【００５５】

実施の形態３．

以上の実施の形態１、２では、運用開始前に外部入力データを利用する汎用ライブラリの呼び出し位置などの情報を静的コード解析で取得し、Ｗｅｂアプリケーションの実行時に運用開始前に取得した情報と、スタックトレースなどの実行時に取得できる情報から、汎用ライブラリの内部で行う脆弱性対策の要否を動的に変更するものである。

これに対して、本実施の形態では、動的試験の試験データもしくはスクリプトを出力する例を示す。

本実施の形態の試験データもしくはスクリプトは、運用開始前の静的コード解析では、脆弱性対策の要否を判定することができないＷｅｂアプリケーションの設定ファイルの誤りによる脆弱性を検出するための動的試験に用いられる。

【００５６】

図７は、本実施の形態における、Ｗｅｂアプリケーション脆弱性対策システム１０００

の構成図である。
図７では、実施の形態２の構成（図５）に対して、開発環境１１００にＷｅｂアプリケーションの設定ファイル１１１１と試験データ／スクリプト１１３１を追加し、さらに静的コード解析部１１２０に、設定ファイル解析部１１２６、試験データ／スクリプト出力部１１２７を追加したものである。
【００５７】
設定ファイル解析部１１２６は、Ｗｅｂアプリケーションの設定ファイル１１１１を読込んで、設定ファイルに誤りがないことの検証と、設定ファイルで指定された値を抽出する。
試験データ／スクリプト出力部１１２７は、静的コード解析部１１２０でのＷｅｂアプリケーションのソース／実行コード１１１０やＷｅｂアプリケーションの設定ファイル１１１１に対する解析結果を用いて、試験データ／スクリプト１１３１を出力する。
なお、本実施の形態に係る開発環境１１００は、プログラム解析装置の例に相当する。
【００５８】
実施の形態３における動作を図８を用いて説明する。
【００５９】
開発環境１１００では、Ｗｅｂアプリケーション入力部１１２１でＷｅｂアプリケーションのソース／実行コード１１１０及びＷｅｂアプリケーションの設定ファイル１１１１が読込まれる（Ｓ４０１）。
次に、汎用ライブラリ利用検証部１１２２が、Ｗｅｂアプリケーションのソース／実行コード１１１０が利用する汎用ライブラリ関数を呼び出しているコードの位置（ライン数など）、呼び出しされる汎用ライブラリ関数の種類（ＨＴＭＬ出力、データベースアクセスなど）を調べた上で、汎用ライブラリ関数に与えるデータや使い方が適切であることを検証する（Ｓ４０２）。
汎用ライブラリ関数が適切に利用されていると判定した場合（Ｓ４０２のＹｅｓ）には、設定ファイル解析部１１２６が、Ｗｅｂアプリケーションの設定ファイル１１１１を解析し、Ｗｅｂアプリケーションの設定ファイル１１１１の設定内容に、設定可能な値の範囲内にないなどの明らかな設定誤りがないかを検証する（Ｓ４０３）。
汎用ライブラリ関数が適切に利用されていない場合（Ｓ４０２のＮｏ）や、Ｗｅｂアプリケーションの設定ファイル１１１１に設定ミスがあると判定した場合（Ｓ４０３のＮｏ）には、適切に利用されていない汎用ライブラリ関数の種類、コード上の位置、適切に利用されていないと判定した理由や、Ｗｅｂアプリケーションの設定ファイル１１１１の設定ミスの内容などを含むメッセージを出力して処理を終了する（Ｓ４０４）。
【００６０】
汎用ライブラリ関数が適切に利用されており、かつＷｅｂアプリケーションの設定ファイル１１１１に設定誤りがないと判定した場合には（Ｓ４０３のＮｏ）、Ｓ４０５～Ｓ４０７の処理が行われる。
これらの処理は、図６のＳ３０３～Ｓ３０６の処理と同じであるため説明は省略する。
【００６１】
次に、試験データ／スクリプト出力部１１２７は、Ｓ４０２、Ｓ４０３、Ｓ４０５で解析した結果に基づいて、汎用ライブラリ関数が利用されている処理でＷｅｂアプリケーションの設定ファイル１１１１の設定によって汎用ライブラリ関数の処理の内容が変更される処理について、動的試験をするための試験データ／スクリプト１１３１を出力する（Ｓ４０８）。
つまり、試験データ／スクリプト出力部１１２７は、Ｗｅｂアプリケーションの設定ファイル１１１の値、脆弱性ライブラリ関数、Ｗｅｂアプリケーションで外部入力データが利用されるコード位置などを用いて、Ｗｅｂアプリケーションの設定ファイル１１１の値の適否を検査するための試験データ又はスクリプトを生成する。
【００６２】
以上のように、本実施の形態では、Ｗｅｂアプリケーション及び設定ファイルを静的コ

ード解析した結果に基づいて、Ｗｅｂアプリケーションの設定ファイルの誤りによる脆弱性を検査するための試験データもしくはスクリプトを作成することによって、静的コード解析では検出できない設定ファイルの誤りを運用開始前に見つけ出すことが容易になる。

【００６３】

以上、本実施の形態では、Ｗｅｂアプリケーションの設定ファイルに依存する静的コード解析で得られた情報だけでは、ライブラリ内部で行う動作を決定できない場合に、設定ファイルに誤りがないかを動的に試験するための試験データやスクリプトを出力する静的コード解析部を有することを説明した。

【００６４】

最後に、実施の形態１～３に示したＷｅｂアプリケーション脆弱性対策システム１０００のハードウェア構成例について説明する。

図９は、実施の形態１～３に示すＷｅｂアプリケーション脆弱性対策システム１０００のハードウェア資源の一例を示す図である。

なお、図９の構成は、あくまでもＷｅｂアプリケーション脆弱性対策システム１０００のハードウェア構成の一例を示すものであり、Ｗｅｂアプリケーション脆弱性対策システム１０００のハードウェア構成は図９に記載の構成に限らず、他の構成であってもよい。

【００６５】

図９において、Ｗｅｂアプリケーション脆弱性対策システム１０００は、プログラムを実行するＣＰＵ９１１を備えている。

ＣＰＵ９１１は、バス９１２を介して、例えば、ＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）９１３、ＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）９１４、通信ボード９１５、表示装置９０１、キーボード９０２、マウス９０３、磁気ディスク装置９２０と接続され、これらのハードウェアデバイスを制御する。

更に、ＣＰＵ９１１は、ＦＤＤ９０４（Ｆｌｅｘｉｂｌｅ Ｄｉｓｋ Ｄｒｉｖｅ）、コンパクトディスク装置９０５（ＣＤＤ）、プリンタ装置９０６、スキャナ装置９０７と接続していてもよい。また、磁気ディスク装置９２０の代わりに、ＳＳＤ（Ｓｏｌｉｄ Ｓｔａｔｅ Ｄｒｉｖｅ）、光ディスク装置、メモリカード（登録商標）読み書き装置などの記憶装置でもよい。

ＲＡＭ９１４は、揮発性メモリの一例である。ＲＯＭ９１３、ＦＤＤ９０４、ＣＤＤ９０５、磁気ディスク装置９２０の記憶媒体は、不揮発性メモリの一例である。これらは、記憶装置の一例である。

実施の形態１～３で説明した「動作設定ファイル記憶部１２４０」は、ＲＡＭ９１４、磁気ディスク装置９２０等により実現される。

通信ボード９１５、キーボード９０２、マウス９０３、スキャナ装置９０７などは、入力装置の一例である。

また、通信ボード９１５、表示装置９０１、プリンタ装置９０６などは、出力装置の一例である。

【００６６】

通信ボード９１５は、図１に示すように、ネットワークに接続されている。

例えば、通信ボード９１５は、ＬＡＮ（ローカルエリアネットワーク）、インターネット、ＷＡＮ（ワイドエリアネットワーク）、ＳＡＮ（ストレージエリアネットワーク）などに接続されていてもよい。

【００６７】

磁気ディスク装置９２０には、オペレーティングシステム９２１（ＯＳ）、ウィンドウシステム９２２、プログラム群９２３、ファイル群９２４が記憶されている。

プログラム群９２３のプログラムは、ＣＰＵ９１１がオペレーティングシステム９２１、ウィンドウシステム９２２を利用しながら実行する。

【００６８】

また、ＲＡＭ９１４には、ＣＰＵ９１１に実行させるオペレーティングシステム９２１のプログラムやアプリケーションプログラムの少なくとも一部が一時的に格納される。

また、ＲＡＭ９１４には、ＣＰＵ９１１による処理に必要な各種データが格納される。

【００６９】

また、ＲＯＭ９１３には、ＢＩＯＳ（Ｂａｓｉｃ　Ｉｎｐｕｔ　Ｏｕｔｐｕｔ　Ｓｙｓｔｅｍ）プログラムが格納され、磁気ディスク装置９２０にはブートプログラムが格納されている。

Ｗｅｂアプリケーション脆弱性対策システム１０００の起動時には、ＲＯＭ９１３のＢＩＯＳプログラム及び磁気ディスク装置９２０のブートプログラムが実行され、ＢＩＯＳプログラム及びブートプログラムによりオペレーティングシステム９２１が起動される。

【００７０】

上記プログラム群９２３には、実施の形態１～３の説明において「～部」（「動作設定ファイル記憶部１２４０」以外、以下同様）として説明している機能を実行するプログラム、「Ｗｅｂアプリケーション１２２０」、「Ｗｅｂサーバプログラム１２１０」が記憶されている。

プログラムは、ＣＰＵ９１１により読み出され実行される。

【００７１】

ファイル群９２４には、実施の形態１～３の説明において、「～の判断」、「～の判定」、「～の抽出」、「～の比較」、「～の検証」、「～の生成」、「～の設定」、「～の取得」、「～の照合」、「～の選択」、「～の生成」、「～の入力」、「～の出力」等として説明している処理の結果を示す情報やデータや信号値や変数値が、ディスクやメモリなどの記憶媒体にファイルとして記憶されている。

また、暗号鍵・復号鍵や乱数値やパラメータが、ディスクやメモリなどの記憶媒体にファイルとして記憶されてもよい。

「～ファイル」や「～データベース」は、ディスクやメモリなどの記憶媒体に記憶される。

ディスクやメモリなどの記憶媒体に記憶された情報やデータや信号値や変数値やパラメータは、読み書き回路を介してＣＰＵ９１１によりメインメモリやキャッシュメモリに読み出される。

そして、読み出された情報やデータや信号値や変数値やパラメータは、抽出・検索・参照・比較・演算・計算・処理・編集・出力・印刷・表示などのＣＰＵの動作に用いられる。

抽出・検索・参照・比較・演算・計算・処理・編集・出力・印刷・表示のＣＰＵの動作の間、情報やデータや信号値や変数値やパラメータは、メインメモリ、レジスタ、キャッシュメモリ、バッファメモリ等に一時的に記憶される。

また、実施の形態１～３で説明しているフローチャートの矢印の部分は主としてデータや信号の入出力を示す。

データや信号値は、ＲＡＭ９１４のメモリ、ＦＤＤ９０４のフレキシブルディスク、ＣＤ９０５のコンパクトディスク、磁気ディスク装置９２０の磁気ディスク、その他光ディスク、ブルーレイ（登録商標）ディスク、ＤＶＤ等の記憶媒体に記録される。

また、データや信号は、バス９１２や信号線やケーブルその他の伝送媒体によりオンライン伝送される。

【００７２】

また、実施の形態１～３の説明において「～部」として説明しているものは、「～回路」、「～装置」、「～機器」であってもよく、また、「～ステップ」、「～手順」、「～処理」であってもよい。

また、Ｗｅｂアプリケーション脆弱性対策システム１０００の処理をプログラム実行方法として捉えることができる。

また、「～部」として説明しているものは、ＲＯＭ９１３に記憶されたファームウェアで実現されていても構わない。

或いは、ソフトウェアのみ、或いは、素子・デバイス・基板・配線などのハードウェアのみ、或いは、ソフトウェアとハードウェアとの組み合わせ、さらには、ファームウェア

との組み合わせで実施されても構わない。
ファームウェアとソフトウェアは、プログラムとして、磁気ディスク、フレキシブルディスク、光ディスク、コンパクトディスク、ブルーレイ（登録商標）ディスク、ＤＶＤ等の記憶媒体に記憶される。
プログラムはＣＰＵ９１１により読み出され、ＣＰＵ９１１により実行される。
すなわち、プログラムは、実施の形態１～３の「～部」としてコンピュータを機能させるものである。あるいは、実施の形態１～３の「～部」の手順や方法をコンピュータに実行させるものである。

【００７３】

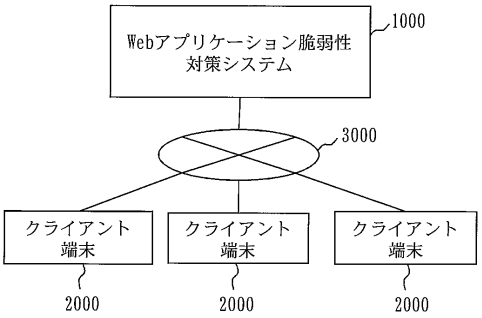
このように、実施の形態１～３に示すＷｅｂアプリケーション脆弱性対策システム１０００は、処理装置たるＣＰＵ、記憶装置たるメモリ、磁気ディスク等、入力装置たるキーボード、マウス、通信ボード等、出力装置たる表示装置、通信ボード等を備えるコンピュータである。
そして、上記したように「～部」として示された機能をこれら処理装置、記憶装置、入力装置、出力装置を用いて実現するものである。

【符号の説明】

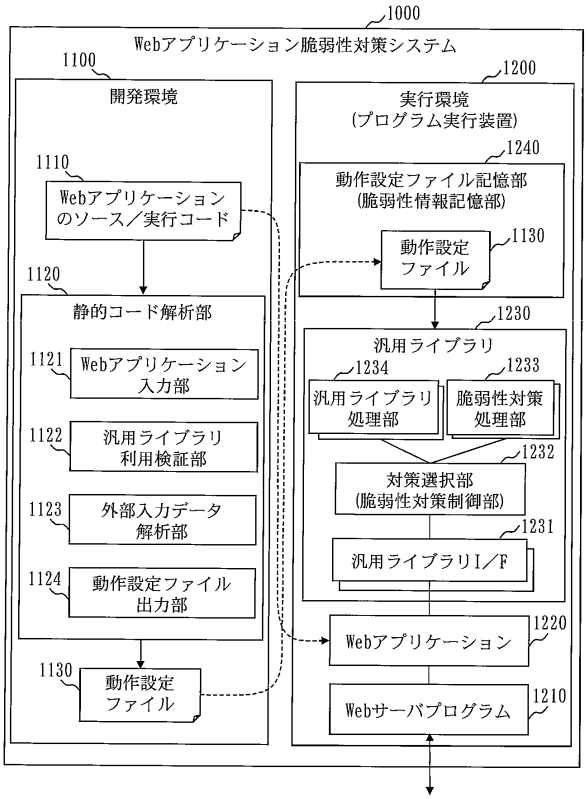
【００７４】

１０００　Ｗｅｂアプリケーション脆弱性対策システム、１１００　開発環境、１１１０　Ｗｅｂアプリケーションのソース／実行コード、１１１１　Ｗｅｂアプリケーションの設定ファイル、１１２０　静的コード解析部、１１２１　Ｗｅｂアプリケーション入力部、１１２２　汎用ライブラリ利用検証部、１１２３　外部入力データ解析部、１１２４　動作設定ファイル出力部、１１２５　構文解析部、１１２６　設定ファイル解析部、１１２７　試験データ／スクリプト出力部、１１３０　動作設定ファイル、１１３１　試験データ／スクリプト、１２００　実行環境、１２１０　Ｗｅｂサーバプログラム、１２２０　Ｗｅｂアプリケーション、１２３０　汎用ライブラリ、１２３１　汎用ライブラリＩ／Ｆ、１２３２　対策選択部、１２３３　脆弱性対策処理部、１２３４　汎用ライブラリ処理部、１２４０　動作設定ファイル記憶部、２０００　クライアント端末、３０００　ネットワーク。

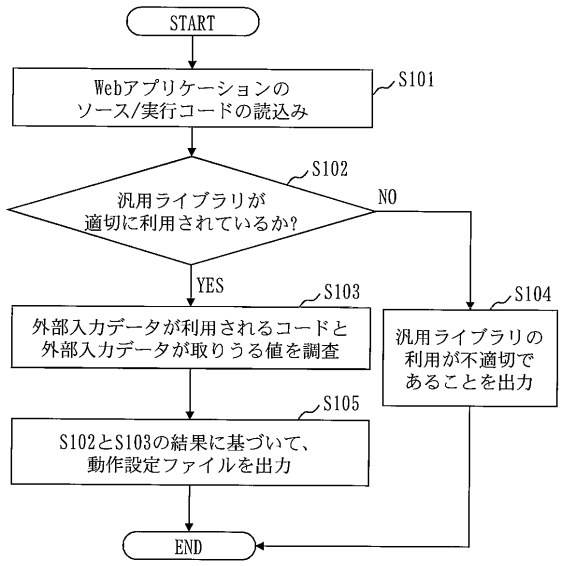
【図１】


Webアプリケーション脆弱性
対策システム
1000
3000
クライアント
端末
クライアント
端末
クライアント
端末
2000
2000
2000

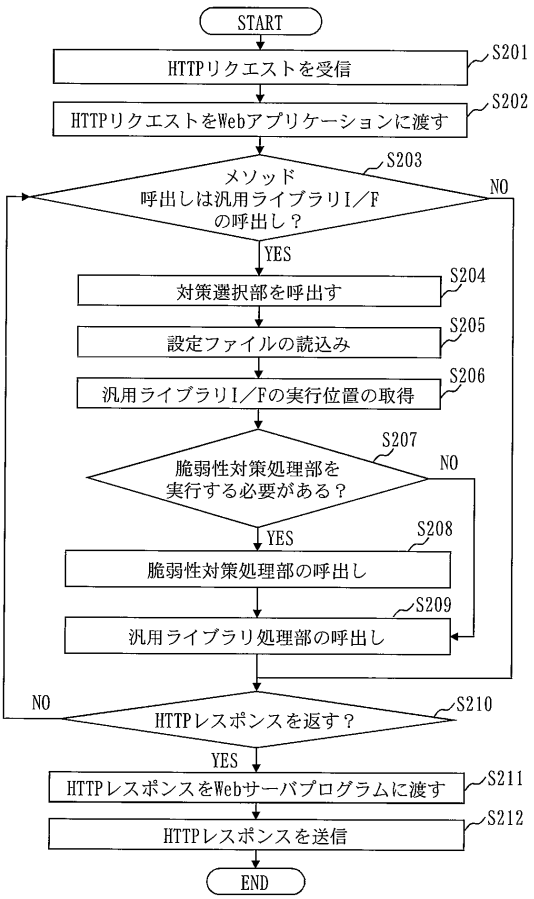
【図２】


1000
Webアプリケーション脆弱性対策システム
1100
開発環境
1200
実行環境
(プログラム実行装置)
1110
Webアプリケーション
のソース／実行コード
1240
動作設定ファイル記憶部
(脆弱性情報記憶部)
動作設定
ファイル
1130
1120
静的コード解析部
1230
汎用ライブラリ
1121
Webアプリケーション
入力部
1234
汎用ライブラリ
処理部
1233
脆弱性対策
処理部
1122
汎用ライブラリ
利用検証部
1232
対策選択部
(脆弱性対策制御部)
1123
外部入力データ
解析部
1231
汎用ライブラリI／F
1124
動作設定ファイル
出力部
1220
Webアプリケーション
1130
動作設定
ファイル
1210
Webサーバプログラム

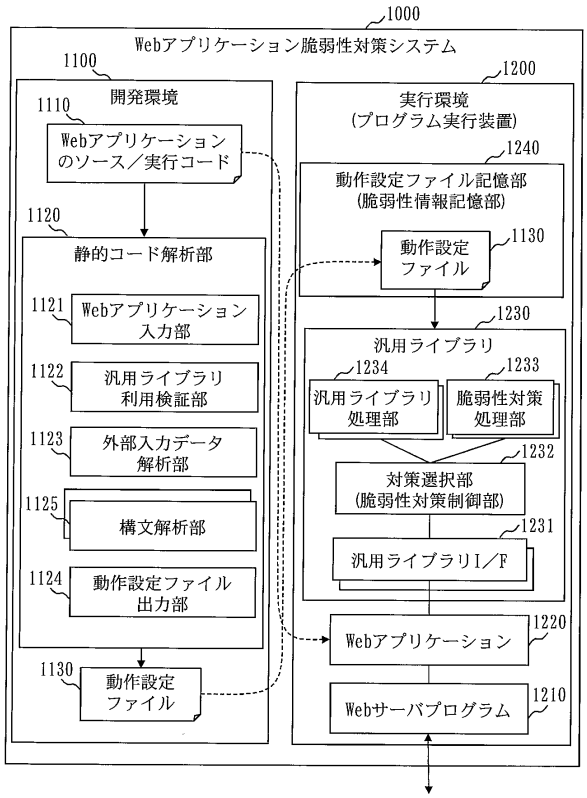
【図３】


START
Webアプリケーションの
ソース/実行コードの読込み
S101
S102
汎用ライブラリが
適切に利用されているか?
NO
YES
S103
外部入力データが利用されるコードと
外部入力データが取りうる値を調査
S104
汎用ライブラリの
利用が不適切で
あることを出力
S105
S102とS103の結果に基づいて、
動作設定ファイルを出力
END

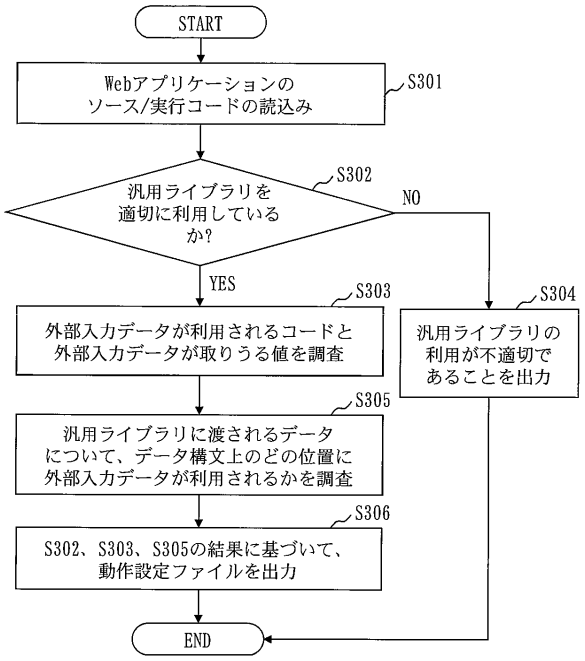
【図４】


START
HTTPリクエストを受信
S201
HTTPリクエストをWebアプリケーションに渡す
S202
S203
メソッド
呼出しは汎用ライブラリI／F
の呼出し？
NO
YES
対策選択部を呼出す
S204
設定ファイルの読込み
S205
汎用ライブラリI／Fの実行位置の取得
S206
S207
脆弱性対策処理部を
実行する必要がある？
NO
YES
S208
脆弱性対策処理部の呼出し
S209
汎用ライブラリ処理部の呼出し
NO
HTTPレスポンスを返す？
S210
YES
HTTPレスポンスをWebサーバプログラムに渡す
S211
HTTPレスポンスを送信
S212
END

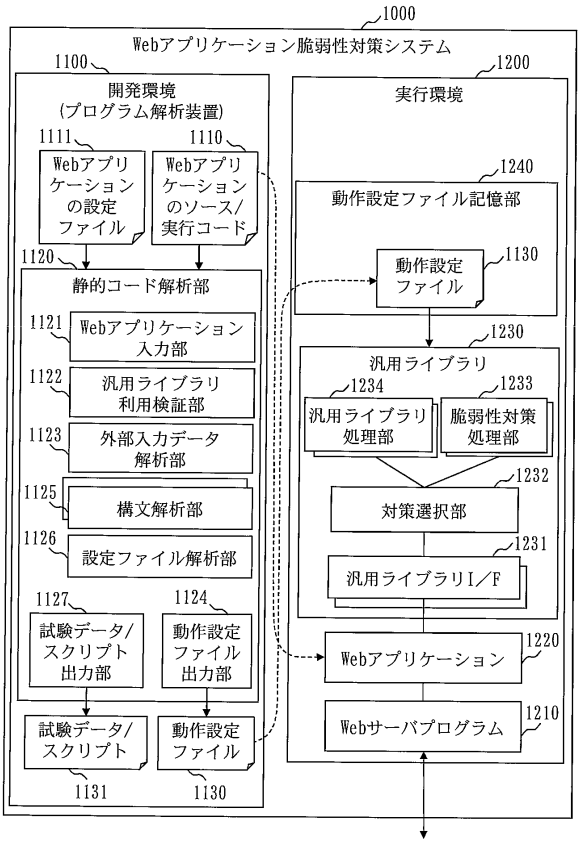
【図５】


1000
Webアプリケーション脆弱性対策システム
1100
開発環境
1110
Webアプリケーション
のソース／実行コード
1120
静的コード解析部
1121
Webアプリケーション
入力部
1122
汎用ライブラリ
利用検証部
1123
外部入力データ
解析部
1125
構文解析部
1124
動作設定ファイル
出力部
1130
動作設定
ファイル
1200
実行環境
(プログラム実行装置)
1240
動作設定ファイル記憶部
(脆弱性情報記憶部)
1130
動作設定
ファイル
1230
汎用ライブラリ
1234
汎用ライブラリ
処理部
1233
脆弱性対策
処理部
1232
対策選択部
(脆弱性対策制御部)
1231
汎用ライブラリI／F
1220
Webアプリケーション
1210
Webサーバプログラム

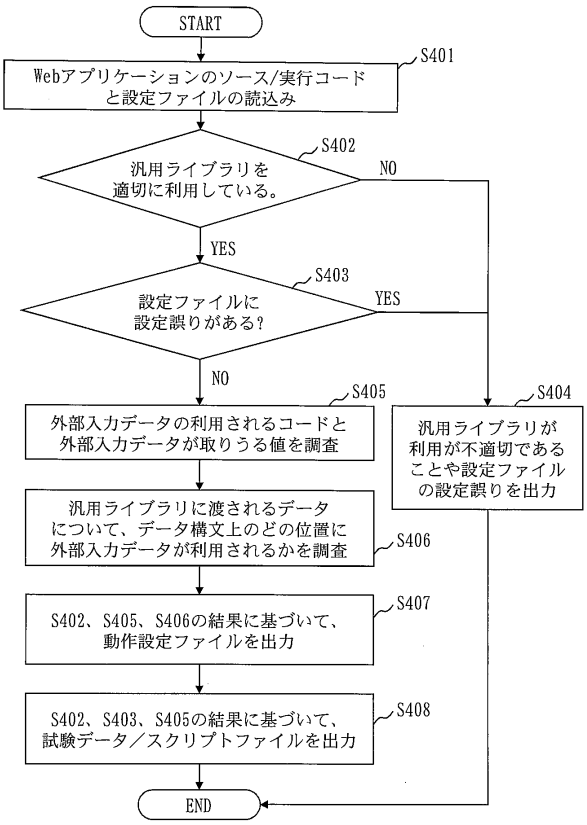
【図６】


START
Webアプリケーションの
ソース/実行コードの読込み
S301
汎用ライブラリを
適切に利用している
か?
S302
NO
YES
外部入力データが利用されるコードと
外部入力データが取りうる値を調査
S303
汎用ライブラリの
利用が不適切で
あることを出力
S304
汎用ライブラリに渡されるデータ
について、データ構文上のどの位置に
外部入力データが利用されるかを調査
S305
S302、S303、S305の結果に基づいて、
動作設定ファイルを出力
S306
END

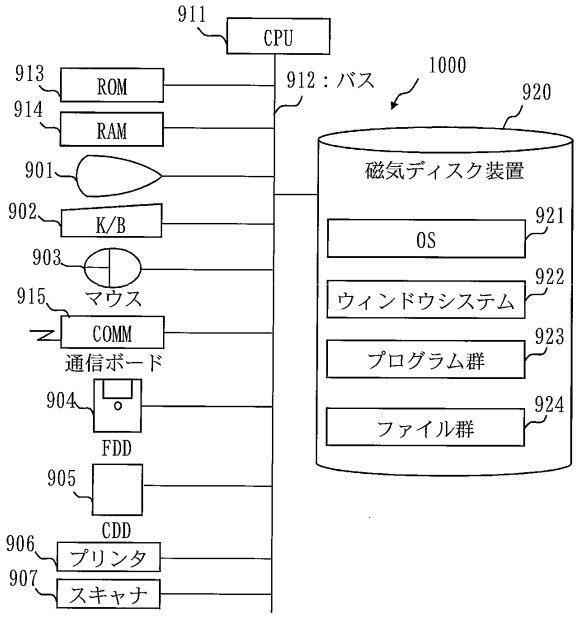
【図７】


1000
Webアプリケーション脆弱性対策システム
1100
開発環境
(プログラム解析装置)
1111
Webアプリ
ケーション
の設定
ファイル
1110
Webアプリ
ケーション
のソース/
実行コード
1120
静的コード解析部
1121
Webアプリケーション
入力部
1122
汎用ライブラリ
利用検証部
1123
外部入力データ
解析部
1125
構文解析部
1126
設定ファイル解析部
1127
試験データ/
スクリプト
出力部
1124
動作設定
ファイル
出力部
試験データ/
スクリプト
1131
動作設定
ファイル
1130
1200
実行環境
1240
動作設定ファイル記憶部
1130
動作設定
ファイル
1230
汎用ライブラリ
1234
汎用ライブラリ
処理部
1233
脆弱性対策
処理部
1232
対策選択部
1231
汎用ライブラリI／F
1220
Webアプリケーション
1210
Webサーバプログラム

【図８】


START
Webアプリケーションのソース/実行コード
と設定ファイルの読込み
S401
汎用ライブラリを
適切に利用している。
S402
NO
YES
設定ファイルに
設定誤りがある?
S403
YES
NO
外部入力データの利用されるコードと
外部入力データが取りうる値を調査
S405
汎用ライブラリが
利用が不適切であるこ
とや設定ファイルの
設定誤りを出力
S404
汎用ライブラリに渡されるデータ
について、データ構文上のどの位置に
外部入力データが利用されるかを調査
S406
S402、S405、S406の結果に基づいて、
動作設定ファイルを出力
S407
S402、S403、S405の結果に基づいて、
試験データ／スクリプトファイルを出力
S408
END

【図９】


911
CPU
913
ROM
914
RAM
912：バス
1000
920
磁気ディスク装置
901
902
K/B
903
マウス
915
COMM
通信ボード
904
FDD
905
CDD
906
プリンタ
907
スキャナ
OS
921
ウィンドウシステム
922
プログラム群
923
ファイル群
924

フロントページの続き

(56)参考文献 特開２００７－４７８８４（ＪＰ，Ａ）
特開２０１２－４８３４２（ＪＰ，Ａ）
特開２０１１－１５０７１６（ＪＰ，Ａ）
特開２０１３－３００１７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　２１／１２