(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6778761号
(P6778761)
(45)発行日　令和2年11月4日(2020.11.4)　(24)登録日　令和2年10月14日(2020.10.14)

(51)Int.Cl.　F I
G06F 21/56 (2013.01)　G06F 21/56 320

請求項の数 18 (全 24 頁)

(21)出願番号　特願2018-552688(P2018-552688)
(86)(22)出願日　平成29年4月6日(2017.4.6)
(65)公表番号　特表2019-514119(P2019-514119A)
(43)公表日　令和1年5月30日(2019.5.30)
(86)国際出願番号　PCT/US2017/026359
(87)国際公開番号　WO2017/177003
(87)国際公開日　平成29年10月12日(2017.10.12)
審査請求日　平成30年11月20日(2018.11.20)
(31)優先権主張番号　62/318,844
(32)優先日　平成28年4月6日(2016.4.6)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　15/479,928
(32)優先日　平成29年4月5日(2017.4.5)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　504080663
エヌイーシー　ラボラトリーズ　アメリカ　インク
ＮＥＣ　Ｌａｂｏｒａｔｏｒｉｅｓ　Ａｍｅｒｉｃａ，　Ｉｎｃ．
アメリカ合衆国　０８５４０　ニュージャージー州　プリンストン　スイート　２００　インディペンデンス　ウェイ　４
(74)代理人　100123788
弁理士　宮崎　昭夫
(74)代理人　100127454
弁理士　緒方　雅昭

最終頁に続く

(54)【発明の名称】ハイブリッドプログラムバイナリ特徴の抽出及び比較

(57)【特許請求の範囲】
【請求項１】
プログラムバイナリの類似度を特定するための方法であって、
１つまたは複数の入力プログラムバイナリからプログラムバイナリ特徴を抽出し、対応するハイブリッド特徴を生成することであって、前記ハイブリッド特徴が、参照特徴、リソース特徴、抽象化制御フロー特徴及び構造特徴を含むことと、
前記抽出されたハイブリッド特徴を用いた複数のバイナリ対の組み合わせを生成することと、
前記ハイブリッド特徴のうち同じ種類の特徴どうしを比較することで、前記バイナリ対毎の類似度スコアを決定することと、
入力されるハイブリッド特徴パラメータと、前記バイナリ対毎の前記類似度スコアとに基づいて、前記バイナリ対における２つの前記プログラムバイナリの類似度を表すハイブリッド差異スコアを生成することと、
既知のマルウェアのプログラムバイナリ及び前記入力プログラムバイナリ間の前記ハイブリッド差異スコアに基づいて、前記入力プログラムバイナリに関する前記マルウェアの尤度を特定することと、
を含む、方法。
【請求項２】
前記複数のバイナリ対は、想定される全ての組み合わせのバイナリ対を含む、請求項１に記載の方法。

【請求項３】
所定の閾値ハイブリッド差異スコアに到達している場合に、マルウェア対策ソフトウェアにおけるマルウェア定義ライブラリを更新することをさらに含む、請求項１に記載の方法。
【請求項４】
前記参照特徴は、入力プログラムから参照されるプログラム及び参照される関数のリストのうちの少なくとも１つを含む、請求項１に記載の方法。
【請求項５】
前記リソース特徴は、グローバルデータ、プログラムメタデータ、プログラムアイコン、文字列及びデバッグシンボルのうちの少なくとも１つを含む、請求項１に記載の方法。
【請求項６】
入力プログラムを逆アセンブリし、命令が制御依存である場合、オペコードのみを取り出し、前記抽象化制御フロー特徴に含める、請求項１に記載の方法。
【請求項７】
前記構造特徴は、バイナリセクションの名称及びバイナリセクションのサイズのリストのうちの少なくとも１つを含む、請求項１に記載の方法。
【請求項８】
前記ハイブリッド特徴パラメータは、前記ハイブリッド差異スコアを生成するときの前記ハイブリッド特徴毎の貢献量を示す所定のレートのセットである、請求項１に記載の方法。
【請求項９】
プログラムバイナリの類似度を特定するためのシステムであって、
メモリと接続されたプロセッサを備え、前記プロセッサは、
１つまたは複数の入力プログラムバイナリからプログラムバイナリ特徴を抽出し、対応するハイブリッド特徴を生成することであって、前記ハイブリッド特徴が、参照特徴、リソース特徴、抽象化制御フロー特徴及び構造特徴を含み、
前記抽出されたハイブリッド特徴を用いた複数のバイナリ対の組み合わせを生成し、
前記ハイブリッド特徴のうち同じ種類の特徴どうしを比較することで、前記バイナリ対毎の類似度スコアを決定し、
入力されるハイブリッド特徴パラメータと、前記バイナリ対毎の前記類似度スコアとに基づいて、前記バイナリ対における２つの前記プログラムバイナリの類似度を表すハイブリッド差異スコアを生成し、
既知のマルウェアのプログラムバイナリ及び前記入力プログラムバイナリ間の前記ハイブリッド差異スコアに基づいて前記入力プログラムバイナリに関する前記マルウェアの尤度を特定するように構成された、システム。
【請求項１０】
前記複数のバイナリ対は、想定される全ての組み合わせのバイナリ対を含む、請求項９に記載のシステム。
【請求項１１】
前記プロセッサは、所定の閾値ハイブリッド差異スコアに到達している場合に、マルウェア対策ソフトウェアにおけるマルウェア定義ライブラリを更新するようにさらに構成された、請求項９に記載のシステム。
【請求項１２】
前記参照特徴は、入力プログラムから参照されるプログラム及び参照される関数のリストのうちの少なくとも１つを含む、請求項９に記載のシステム。
【請求項１３】
前記リソース特徴は、グローバルデータ、プログラムメタデータ、プログラムアイコン、文字列及びデバッグシンボルのうちの少なくとも１つを含む、請求項９に記載のシステム。
【請求項１４】

前記プロセッサは、入力プログラムを逆アセンブリし、命令が制御依存である場合、オペコードのみを取り出し、前記抽象化制御フロー特徴に含める、請求項９に記載のシステム。

【請求項１５】

前記構造特徴は、バイナリセクションの名称及びバイナリセクションのサイズのリストのうちの少なくとも１つを含む、請求項９に記載のシステム。

【請求項１６】

前記ハイブリッド特徴パラメータは、前記ハイブリッド差異スコアを生成するときの前記ハイブリッド特徴毎の貢献量を示す所定のレートのセットである、請求項９に記載のシステム。

【請求項１７】

プログラムバイナリの類似度を特定するためのコンピュータで読み取り可能なプログラムを備えた非一時的なコンピュータで読み取り可能な記憶媒体であって、

前記コンピュータで読み取り可能なプログラムは、

１つまたは複数の入力プログラムバイナリからプログラムバイナリ特徴を抽出し、対応するハイブリッド特徴を生成する工程であって、前記ハイブリッド特徴が、参照特徴、リソース特徴、抽象化制御フロー特徴及び構造特徴を含み、

前記抽出されたハイブリッド特徴を用いた複数のバイナリ対の組み合わせを生成する工程と、

前記ハイブリッド特徴のうち同じ種類の特徴どうしを比較することで、前記バイナリ対毎の類似度スコアを決定する工程と、

入力されるハイブリッド特徴パラメータと、前記バイナリ対毎の前記類似度スコアとに基づいて、前記バイナリ対における２つの前記プログラムバイナリの類似度を表すハイブリッド差異スコアを生成する工程と、

既知のマルウェアのプログラムバイナリ及び前記入力プログラムバイナリ間の前記ハイブリッド差異スコアに基づいて前記入力プログラムバイナリに関する前記マルウェアの尤度を特定する工程と、

を前記コンピュータに実行させる、非一時的なコンピュータで読み取り可能な記憶媒体。

【請求項１８】

所定の閾値ハイブリッド差異スコアに到達している場合に、マルウェア対策ソフトウェアにおけるマルウェア定義ライブラリを更新する工程をさらに含む、請求項１７に記載の非一時的なコンピュータで読み取り可能な記憶媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

この出願は、２０１６年４月６日に出願された米国仮特許出願第６２／３１８，８４４号を基礎とする優先権を主張し、その開示の全てをここに取り込む。

【０００２】

本発明は、プログラム特徴（program feature）の抽出及び比較に関し、特にハイブリッドプログラムバイナリ特徴の抽出及び比較による、悪意のあるソフトウェア攻撃の検出及び防止に関する。

【背景技術】

【０００３】

プログラムバイナリは、プログラムの特性を理解するためのサイバーセキュリティの重要な態様である。無害な（benign）ソフトウェア及びマルウェアはプログラムバイナリとして配信される。それらの配信及びランタイム動作を調査することは、ウィルス対策ソフトウェア等の多くのサイバーセキュリティソリューションで実行される重要なタスクである。

【０００４】

従来のウィルス対策製品は、バイナリシグネチャを用いてマルウェアを特定し、プログ

ラムバイナリを一意に識別するためにハッシュ値が一般的に用いられてきた。これは、コンテンツ（例えば、バイナリ全体、あるいは特定の１つまたは複数のセクション）に基づく１つの特徴として分類できる。しかしながら、バイナリコンテンツのハッシュ値は、小さな変化に対して敏感であり、バイナリにおける単一ビットの差異であっても、完全に異なるハッシュ値が生じるため、そのようなシステム及び方法を用いると、正確または信頼できる結果が得られない。さらに、バイナリコンテンツにおける差異量は、ハッシュ値の差異ではうまく表せない。

【０００５】

従来の他の手法では、マルウェアファミリーの類似度を判定する試みにおいて、プログラムの制御フロー情報（例えば、ＣＰＵ命令、システムコール）が用いられてきた。しかしながら、これらの手法は、少なくとも部分的に、マルウェアだけでなく無害なソフトウェアにおける感度に起因して、類似度比較において有効でない。したがって、当該方法は、プログラムの類似度を正確に判定するのに有効ではない。

【０００６】

無害なソフトウェアは、異なるプラットフォーム及びパッチのために多くのバージョンを有する。さらに、それらのソースコードが非常に類似している場合でも、一旦、コンパイルされてバイナリフォーマットになると、その命令構造はコンパイラのアルゴリズム及び最適化に起因して大幅に異なるものになる。さらに、悪意のあるソフトウェア（マルウェア）に関して、マルウェアの作成者は、コードが変形したもの（例えば、ポリモーフィックマルウェアコード）を用いる。これは事実上、上述したような従来の手法を混乱させて不正確で無効なものにする。このため、従来の手法は、例えば類似する無害なプログラムとマルウェアファミリーとを正確に判定するには十分に信頼できるものでは無く、有効でもないため、プログラムバイナリの確実で有効な特性解析及び類似度比較は未解決の課題である。

【発明の概要】

【０００７】

本原理の一態様によれば、１つまたは複数の入力プログラムバイナリからプログラムバイナリ特徴を抽出し、対応するハイブリッド特徴を生成することを含む、プログラムバイナリの類似度を特定するための方法が提供される。ハイブリッド特徴は、参照特徴（reference feature）、リソース特徴（resource feature）、抽象化制御フロー特徴（abstract control flow feature）及び構造特徴（structural feature）を含む。抽出されたハイブリッド特徴を用いた複数のバイナリ対の組み合わせを生成し、前記ハイブリッド特徴のうち同じ種類の特徴どうしを比較することで、バイナリ対毎の類似度スコアを決定する。入力されるハイブリッド特徴パラメータと、バイナリ対毎の類似度スコアとに基づいてバイナリ対における２つのプログラムバイナリの類似度を表すハイブリッド差異スコアを生成する。既知のマルウェアのプログラムバイナリ及び入力プログラムバイナリ間のハイブリッド差異スコアに基づいて入力プログラムに関するマルウェアの尤度を特定する。

【０００８】

本原理の他の態様によれば、プログラムバイナリの類似度を特定するためのシステムが提供される。システムは、１つまたは複数の入力プログラムバイナリからプログラムバイナリ特徴を抽出し、対応するハイブリッド特徴を生成するように構成された、メモリと接続されたプロセッサを備える。ハイブリッド特徴は、参照特徴、リソース特徴、抽象化制御フロー特徴及び構造特徴を含む。抽出されたハイブリッド特徴を用いた複数のバイナリ対の組み合わせを生成し、前記ハイブリッド特徴のうち同じ種類の特徴どうしを比較することで、バイナリ対毎の類似度スコアを決定する。入力されるハイブリッド特徴パラメータと、バイナリ対毎の類似度スコアとに基づいてバイナリ対における２つのプログラムバイナリの類似度を表すハイブリッド差異スコアを生成する。既知のマルウェアのプログラムバイナリ及び入力プログラムバイナリ間のハイブリッド差異スコアに基づいて入力プログラムバイナリに関するマルウェアの尤度を特定する。

【０００９】

本原理の他の態様によれば、１つまたは複数の入力プログラムバイナリからプログラムバイナリ特徴を抽出し、対応するハイブリッド特徴を生成することを含む、プログラムバイナリの類似度を特定するための非一時的なコンピュータで読み取り可能な媒体が提供される。ハイブリッド特徴は、参照特徴、リソース特徴、抽象化制御フロー特徴及び構造特徴を含む。抽出されたハイブリッド特徴を用いた複数のバイナリ対の組み合わせを生成し、前記ハイブリッド特徴のうち同じ種類の特徴どうしを比較することで、バイナリ対毎の類似度スコアを決定する。入力されるハイブリッド特徴パラメータと、バイナリ対毎の類似度スコアとに基づいてバイナリ対における２つのプログラムバイナリの類似度を表すハイブリッド差異スコアを生成する。既知のマルウェアのプログラムバイナリ及び入力プログラムバイナリ間のハイブリッド差異スコアに基づいて入力プログラムに関するマルウェアの尤度を特定する。

【００１０】

これら及び他の特徴並びに利点は、以下の典型的な実施形態の詳細な説明を添付の図面と併せて読むことで明らかになるであろう。

【図面の簡単な説明】

【００１１】

本開示では、後述するように、以下の図面を参照しながら好ましい実施形態について詳細に説明する。

【００１２】

【図１】図１は、本原理による、本原理を適用できる典型的な処理システムを示すブロック／フロー図である。

【００１３】

【図２Ａ】図２Ａは、本原理による、プログラムバイナリ特徴抽出のための高レベルのシステム／方法を示すブロック／フロー図である。

【００１４】

【図２Ｂ】図２Ｂは、本原理による、ハイブリッド特徴類似度解析のための高レベルのシステム／方法を示すブロック／フロー図である。

【００１５】

【図３】図３は、本原理による、プログラムバイナリ特徴抽出のための方法を示すブロック／フロー図である。

【００１６】

【図４】図４は、本原理による、プログラムバイナリ特徴抽出のための典型的な参照特徴を示すブロック／フロー図である。

【００１７】

【図５】図５は、本原理による、抽象化制御特徴の生成のための方法を示すブロック／フロー図である。

【００１８】

【図６】図６は、本原理による、ハイブリッド特徴類似度比較のための方法を示すブロック／フロー図である。

【００１９】

【図７】図７は、本原理による、２つのバイナリの類似度比較のための方法を示すブロック／フロー図である。

【００２０】

【図８】図８は、本原理による、ハイブリッドプログラムバイナリ特徴の抽出及び比較のためのシステムを示すブロック／フロー図である。

【発明を実施するための形態】

【００２１】

本原理によれば、ハイブリッドバイナリプログラム特徴の抽出及び比較のためのシステム及び方法が提供される。

【００２２】

特に有用な実施形態において、本原理によるハイブリッドバイナリプログラム特徴の抽出及び比較により、１つまたは複数のコンピュータシステムに対する悪意のあるソフトウェア（マルウェア）攻撃を検出及び／または防止するためのシステム及び方法が提供される。

【００２３】

一実施形態において、本原理は、例えばウィルス対策保護システムのための総合的なウィルス定義アップデータとして、マルウェア攻撃から１つまたは複数のコンピューティングシステムを保護する実用的な解決策として用いることができる。プログラムバイナリは、プログラムの特性を理解するためのサイバーセキュリティの重要な態様である。無害なソフトウェア及びマルウェアは、プログラムバイナリとして配信される。それらの配信及びランタイム動作を調査することは、多くのサイバーセキュリティソリューション（例えば、ウィルス対策ソフトウェア）で実行される重要なタスクである。

【００２４】

現代のコンピューティングシステムは、多くのプログラムを使用し、コンピュータシステムの複雑性は、しばしば複数の機能を実装するために、一緒に動作させる複数のプログラムが必要であることがよく知られている。さらに、例えば多様なオペレーティングシステム及びプラットフォームに起因して、同じソフトウェアが多くの異なるバージョンとしてパッケージ化されて配信されることがある。各オペレーティングシステムは、複数のバージョン（例えば、ＷｉｎｄｏｗｓにおけるＳｅｒｖｉｃｅ　Ｐａｃｋ、Ｌｉｎｕｘにおける配信バージョン等）を有することがある。さらに、プログラムは、多数のエラー及びセキュリティ脆弱性に起因して、頻繁に再コンパイルされる、またはパッチが適用される。例えば、現代のオペレーティングシステムは、複数の理由のうちのいずれかで、プログラムの新たな配信バージョン及び更新バージョンがしばしば発行される。

【００２５】

これらの理由により、企業環境において日々多数のプログラムバイナリが配備及び更新される。これは、サイバーセキュリティシステム及び方法にとって、プログラムの監視及び解析が困難な課題になる。ウィルス対策会社は、新たなマルウェアを解析し、それらのバイナリシグネチャデータベースを更新するために多くの労力及びリソースを集中させている。しかしながら、バイナリベースのシグネチャを解析して分類する既存の手法は、例えばそれらの低い信頼性及びバイナリの規模に起因して、多数のプログラムバイナリの取り扱いにおける困難を含む、様々な問題を被る。

【００２６】

上述したように、従来のシステム及び方法は、プログラム間の類似度または無害なプログラムとマルウェアファミリーとの正確な類似度を有効にまたは確実に判定しない。無害なソフトウェアは、異なるプラットフォーム及びパッチのために多くのバージョンを有する。それらのソースコードが非常に類似している場合でも、一旦、コンパイルされてバイナリフォーマットになると、その命令構造は、例えばコンパイラのアルゴリズム及び最適化に起因して大幅に異なるものになる。

【００２７】

さらに、悪意のあるソフトウェア（マルウェア）に関して、マルウェアの作成者は、コードが変形したもの（例えば、ポリモーフィックマルウェアコード）を用いる。これは事実上、上述したような従来の手法を混乱させて不正確で無効なものにする。このため、上述したような従来の手法は、例えば類似する無害なプログラムとマルウェアファミリーとを正確にかつ確実に判定するために、十分に信頼できるものでは無く、効果があるものでも無いため、プログラムバイナリの確実で有効な特性解析及び類似度比較は未解決の課題である。

【００２８】

様々な実施形態によれば、本原理は、プログラムバイナリから複数の特徴を抽出し、プログラムの特性を数値化して、それらの類似度をブラックボックス手法で（例えば、ソースコードまたはデバッグ情報を何も用いず）比較することに適用できる。特に、本原理に

よれば、抽出される複数の特徴は、（１）参照特徴、（２）リソース特徴、（３）抽象化制御フロー特徴及び／または（４）構造特徴を含む。これらの特徴は、参照されるバイナリ、リソース、制御フロー及びバイナリ構造の観点から見て、バイナリの複数の態様を表している。さらに、これらの特徴は、バイナリの他の特徴よりも関連する特性の適用範囲（coverage）がより豊富であり、相補的にプログラムの類似度を数値化するためにより有効である。

【００２９】

いくつかの実施形態において、本原理は、サイバーセキュリティシステムにおけるプログラムのホワイトリストへの登録及びマルウェアのクラスタリングのための重要な特徴である、プログラムバイナリの有効で正確な比較を可能にするために適用できる。

【００３０】

プログラムのホワイトリストへの登録に関して、本原理による既知の無害なプログラムとの類似度を判定することは、無害なプログラムの多様な変形に対する、複雑度を低減して、マルウェア検出の正確性及び有効性を増大させるために有用である。以下でさらに詳細に示すように、本発明の複数の特徴は、類似度計算のプロセスを改善する。

【００３１】

マルウェアのクラスタリングに関して、大量の新たなマルウェアが毎日のように新たにリリースされ、発見されている。このマルウェアの全てを手作業で調べることは、人力のコストが高く、人力及び／または従来のウィルス対策システムを用いるタイムリーな方法（例えば、絶え間なく変化するマルウェアに対処し、マルウェア変形からの攻撃をリアルタイムで防止するために十分にタイムリーに）では、膨大な量の潜在的なマルウェアを処理するには能力が欠如しており、不的確であるため、非常に困難である。本発明は、以下でさらに詳細に説明する様々な実施形態による、マルウェアの比較及びクラスタリングの品質、有効性及び正確性を改善するのに有利である。

【００３２】

本明細書に記載される実施形態は、全てハードウェアで実現してもよく、全てソフトウェアで実現してもよく、ハードウェアとソフトウェアの両方の要素を含んでいてもよい。好ましい実施形態において、本発明は、ファームウェア、常駐ソフトウェア、マイクロコード等を含むが、これらに限定されないソフトウェアでも実現可能である。

【００３３】

実施形態には、コンピュータもしくは任意の命令実行システムによって使用される、または関連して使用されるプログラムコードを提供する、コンピュータで利用可能な、またはコンピュータで読み取り可能な媒体からアクセスできる、コンピュータプログラム製品を含んでもよい。コンピュータで利用可能な、またはコンピュータで読み取り可能な媒体には、命令実行システム、機器、もしくは装置によって使用される、または関連して使用されるプログラムを格納、伝達、伝搬または転送する任意の機器を含んでもよい。該媒体は、磁気媒体、光学媒体、電子媒体、電磁気媒体、赤外線媒体、または半導体システム（または機器もしくは装置）、あるいは伝搬媒体であってよい。該媒体には、半導体または固体メモリ、磁気テープ、取り外し可能なコンピュータディスケット、ランダムアクセスメモリ（ＲＡＭ）、リードオンリメモリ（ＲＯＭ）、リジッド磁気ディスク及び光ディスク等のコンピュータで読み取り可能な媒体を含んでもよい。

【００３４】

各コンピュータプログラムは、汎用または特別な目的を持つプログラム可能なコンピュータで読み取ることができる、機械で読み取り可能なストレージメディアまたは装置（例えば、プログラムメモリまたは磁気ディスク）に格納される。該コンピュータプログラムは、ストレージメディアまたは装置から本明細書に記載された手順を実行するコンピュータで読み出される、該コンピュータの設定及び制御動作のためのものである。本発明のシステムには、本明細書に記載した機能を実行する、特定の及び事前に定義された方法でコンピュータを動作させるように構成されたコンピュータプログラムを含む、コンピュータで読み取り可能なストレージメディアも考慮される。

【００３５】
プログラムコードを記憶及び／または実行するのに適したデータ処理システムは、システムバスを介してメモリ要素に直接または間接的に接続された少なくとも１つのプロセッサを備えていてもよい。このメモリ要素には、処理の実行中にバルク記憶装置からコードが検索される回数を減らすために、プログラムコードの実際の実行中に用いられるローカルメモリ、バルク記憶装置及び少なくともいくつかのプログラムコードを一時的に記憶するキャッシュメモリを備えていてもよい。入力／出力すなわちＩ／Ｏ装置（限定されるものではないが、キーボード、ディスプレイ、ポインティング装置等を含む）は、直接またはＩ／Ｏコントローラを介してシステムに接続されてもよい。
【００３６】
ネットワークアダプタは、データ処理システムが、プライベートネットワークまたは公衆ネットワークを介して、他のデータ処理システムまたは遠隔プリンタもしくは記憶装置に接続されることを可能にするために、上記システムと接続されていてもよい。モデム、ケーブルモデム及びイーサネット（登録商標）カードは、現在利用可能なタイプのネットワークアダプタのほんの一握りのものである。
【００３７】
ここで、同じ数字が同一または同様の要素を表す図面、まず図１を詳細に参照すると、図１には本原理の一実施形態による、本原理が適用できる典型的な処理システム１００が示されている。
【００３８】
処理システム１００は、システムバス１０２を介して他のコンポーネントと動作可能に接続された、少なくとも１つのプロセッサ（ＣＰＵ）１０４を含む。システムバス１０２には、キャッシュ１０６、リードオンリメモリ（ＲＯＭ）１０８、ランダムアクセスメモリ（ＲＡＭ）１１０、入力／出力（Ｉ／Ｏ）アダプタ１２０、サウンドアダプタ１３０、ネットワークアダプタ１４０、ユーザインタフェースアダプタ１５０及びディスプレイアダプタ１６０が動作可能に接続されている。
【００３９】
第１の記憶装置１２２及び第２の記憶装置１２４は、Ｉ／Ｏアダプタ１２０によってシステムバス１０２に動作可能に接続されている。記憶装置１２２及び１２４は、ディスク記憶装置（例えば、磁気ディスク記憶装置または光ディスク記憶装置）、固体磁気装置等のいずれであってもよい。記憶装置１２２及び１２４は、同じタイプの記憶装置であってもよく、異なるタイプの記憶装置であってもよい。
【００４０】
スピーカ１３２は、サウンドアダプタ１３０によってシステムバス１０２に動作可能に接続されている。トランシーバ１４２は、ネットワークアダプタ１４０によってシステムバス１０２に動作可能に接続されている。ディスプレイ装置１６２は、ディスプレイアダプタ１６０によってシステムバス１０２に動作可能に接続されている。
【００４１】
第１のユーザ入力装置１５２、第２のユーザ入力装置１５４及び第３のユーザ入力装置１５６は、ユーザインタフェースアダプタ１５０よってシステムバス１０２に動作可能に接続されている。ユーザ入力装置１５２、１５４及び１５６は、キーボード、マウス、キーパッド、イメージキャプチャ装置、モーション感知装置、マイクロフォン、あるいはこれらの装置のうちの少なくとも２つの装置の機能を組み込んだ装置等のいずれであってもよい。本原理の趣旨を維持する限りにおいて、他のタイプの入力装置を使用することも可能である。ユーザ入力装置１５２、１５４及び１５６は、同じタイプのユーザ入力装置であってもよく、異なるタイプのユーザ入力装置であってもよい。ユーザ入力装置１５２、１５４及び１５６は、システム１００に情報を入力し、システム１００から情報を出力するために使用される。
【００４２】
処理システム１００は、当業者であれば容易に思いつくような他の要素（図示せず）を

含んでいてもよく、特定の要素を省略することも可能である。例えば、当業者であれば容易に理解できるが、処理システム１００には、その詳細な実装に応じて他の様々な入力装置及び／または出力装置を使用できる。例えば、無線及び／または有線による様々なタイプの入力装置及び／または出力装置を使用できる。さらに、当業者であれば容易に理解できるが、様々な構成において追加のプロセッサ、コントローラ、メモリ等を使用することも可能である。処理システム１００の上記及び他の変形例は、本明細書で提供される本原理の教示によって当業者であれば容易に考えられるであろう。

【００４３】

さらに、図１、図２Ａ、図２Ｂ及び図８に関して説明するシステム１００、２００、２１０及び８００は、本原理の各実施形態を実現するためのシステムであることを理解されたい。本原理の様々な実施形態によれば、処理システム１００の一部または全ては、システム２００、２１０及び８００の要素の１つまたは複数を実行してもよい。

さらに、処理システム１００は、例えば、図２Ａ、図２Ｂ、図３、図４、図５、図６及び図７の方法２００、２１０、３００、４００、５００、６００及び７００の少なくとも一部を含む、本明細書で説明する方法の少なくとも一部を実行してもよいことを理解されたい。同様に、本原理の様々な実施形態によれば、システム８００の一部または全ては、図２Ａ、図２Ｂ、図３、図４、図５、図６及び図７の方法２００、２１０、３００、４００、５００、６００及び７００の少なくとも一部を実行するために使用されてもよい。

【００４４】

次に図２Ａを参照すると、図２Ａには本原理の実施形態によるプログラムバイナリ特徴抽出のための高レベルの方法２００が例示的に示されている。

【００４５】

一実施形態において、ブロック２０２にて、１つまたは複数のプログラムバイナリが入力される。本原理によれば、ブロック２０４において、ハイブリッドバイナリ特徴を生成するためにプログラムバイナリ特徴抽出が実行され、ブロック２０６において、ハイブリッドバイナリ特徴が出力される。

【００４６】

次に図２Ｂを参照すると、図２Ｂには本原理の実施形態によるハイブリッド特徴類似度解析のための高レベルの方法２１０が例示的に示されている。

【００４７】

一実施形態において、ブロック２１２、２１４及び／または２１６において、１つまたは複数の生成されたハイブリッドバイナリ特徴が入力され、ブロック２１８において、１つまたは複数のハイブリッド特徴パラメータが入力される。本原理によれば、ブロック２２０において、２つ以上のプログラムバイナリの類似度を判定するために、特徴２１２、２１４及び２１６、並びに特徴パラメータ２１８を用いることができる。ブロック２２２において、ハイブリッド特徴類似度解析２２０に基づく類似度ベクトルが出力される。様々な実施形態によれば、ブロック２２２で出力される類似度ベクトルは、例えばマルウェア攻撃の検出及び防止のために、リアルタイムな悪意のあるソフトウェア（マルウェア）定義の比較及び更新を提供するために用いることができる。

【００４８】

次に図３を参照すると、図３には本原理の実施形態によるプログラムバイナリ特徴抽出のための方法３００が例示的に示されている。

【００４９】

一実施形態において、本原理によれば、ブロック３０２において、１つまたは複数のプログラムバイナリが入力され、ブロック３０４において、プログラムバイナリ特徴が抽出される。様々な実施形態によれば、プログラムバイナリ３１４で抽出されるハイブリッドバイナリ特徴には、参照特徴３１６、リソース特徴３１８、抽象化制御フロー特徴３２０及び構造特徴３２２のうちの１つまたは複数が含まれる。本原理によれば、参照特徴抽出３０６、リソース特徴抽出３０８、抽象化制御フロー特徴抽出３１０及び構造特徴抽出３１２を含む、対応する抽出機能を用いてハイブリッド特徴が抽出される。様々な実施形態

によれば、出力は、個々の特徴としてハイブリッド特徴で保存される。

【００５０】

説明を容易にするため、本システム及び方法の様々な特徴に関して、以下の表現を用いる。

Ｐ：入力プログラムバイナリ３０２

Ｆ＿Ｆ（Ｐ）：プログラムバイナリＰの参照特徴３１６

Ｆ＿Ｒ（Ｐ）：プログラムバイナリＰのリソース特徴３１８

Ｆ＿Ｃ（Ｐ）：プログラムバイナリＰの抽象化制御特徴３２０

Ｆ＿Ｓ（Ｐ）：プログラムバイナリＰの構造特徴３２２

【００５１】

一実施形態において、プログラムＰのためのハイブリッドバイナリ特徴３１４は、以下で示すように、プログラムＰのための参照特徴、リソース特徴、抽象化制御特徴及び構造特徴の４タプルである。

ＨＦ（Ｐ）＝（Ｆ＿Ｆ（Ｐ），Ｆ＿Ｒ（Ｐ），Ｆ＿Ｃ（Ｐ），Ｆ＿Ｓ（Ｐ））

【００５２】

ハイブリッド特徴の類似度は、以下のように表せばよい。

【００５３】

プログラムＰ１、Ｐ２の参照特徴３１６の類似度：

Ｓｉｍ＿Ｆ（Ｆ＿Ｆ（Ｐ１），Ｆ＿Ｆ（Ｐ２））

その入力は、プログラムＰ１及びＰ２の２つの参照特徴である。

この値は、０（０％）と１（１００％）との間である。

【００５４】

プログラムＰ１、Ｐ２のリソース特徴３１８の類似度：

Ｓｉｍ＿Ｒ（Ｆ＿Ｒ（Ｐ１），Ｆ＿Ｒ（Ｐ２））

その入力は、プログラムＰ１及びＰ２の２つのリソース特徴である。

この値は、０（０％）と１（１００％）との間である。

【００５５】

プログラムＰ１、Ｐ２の抽象化制御フロー特徴３２０の類似度：

Ｓｉｍ＿Ｃ（Ｆ＿Ｃ（Ｐ１），Ｆ＿Ｃ（Ｐ２））

その入力は、プログラムＰ１及びＰ２の２つの抽象化制御フロー特徴である。

この値は、０（０％）と１（１００％）との間である。

【００５６】

プログラムＰ１、Ｐ２の構造特徴３２２の類似度：

Ｓｉｍ＿Ｃ（Ｆ＿Ｃ（Ｐ１），Ｆ＿Ｃ（Ｐ２））

その入力は、プログラムＰ１及びＰ２の２つの構造特徴である。

この値は、０（０％）と１（１００％）との間である。

【００５７】

様々な実施形態によれば、プログラムバイナリ特徴抽出３０４は、以下のように実行すればよい。

【数１】

Given the following:

P: an input program binary

F_F(P): Reference feature for a program P

F_R(P): Resource feature for a program P

F_C(P): Abstract control feature for a program P

F_S(P): Structual feature for a program P

【００５８】
一実施形態において、参照特徴抽出３０６は、以下のように実行すればよい。
【数２】

```
Extract_F_F(P) // Extract Reference Feature from a Program binary P
        ListOfReferences = [] // empty set
        Sections = Get the list of binary sections of Program P
        For each section in Sections:
            If section is a reference table (For example, import Table, GOT in ELF)
                For each entry in section:
                        program = getReferredProgram(entry)
                        function = getReferredFunctionName(entry)
                        ListOfReferences.add(pair(program, function))
        Return ListOf References // This is F_F(P)
```

【００５９】
一実施形態において、リソース特徴抽出３０８は、以下のように実行すればよい。
【数３】

```
Extract_F_R(P) // Extract Resource Feature from a Program binary P
        ListOfResource = [] // empty set
        Sections = Get the list of binary sections of Program P
        For each section in Sections:
            If section has resource (E.g., String, Symbol, Global Data, Icon, etc.)
                    kind = getKind(section)
                    For each entry in section:
                            value = getValue(entry)
                            ListOfResource.add(pair(kind, value))
        Return ListOfResource // This is F_R(P)
```

【００６０】
一実施形態において、抽象化制御特徴抽出３１０は、以下のように実行すればよい。

【数４】

```
Extract_F_C(P) // Extract Abstract Control Flow Feature from a Program binary P
        ListOfACF = [] // empty set
        Sections = Get the list of binary sections of Program P
        For each section in Sections:
            If section is a code section
                DisassembledInstructions = Disassemble(section)
                For each instruction in DisasembledInstructions:
                    If instruction is control dependent
                        ACF = getOpCode(instruction)
                        ListOfACF. add(ACF)
        Return ListOfACF // This is F_C(P)
```

【００６１】

一実施形態において、構造特徴抽出３１２は、以下のように実行すればよい。

【数５】

```
Extract_F_S(P) // Extract Structual Feature from a Program binary P
        ListOfSections = [] // empty set
        Sections = Get the list of binary sections of Program P
        For each section in Sections:
            name = getSectionName(section)
            size = getSectionSize(section)
            ListOfSections. add(pair(name, size))
        Return ListOfSections // This is F_S(P)
```

【００６２】

本原理によれば、上記特徴の４タプルは、少なくとも参照特徴がライブラリとの関係を示し、リソース特徴が同じプログラムの異なる複数のバージョンまたは類似する複数のプログラムで共有できる共通の文字列またはメッセージを示し、抽象化制御フロー特徴がより少ないノイズを有するプログラム命令の類似度を示し、構造特徴がセクション情報を比較することでプログラム構造全体の類似度を示すという理由で用いられるため、例えば本原理によるマルウェア攻撃を防止するためのプログラム間の類似度の検出に有用である。

【００６３】

次に図４を参照すると、図４には本原理によるプログラムバイナリ特徴抽出のための典型的な参照特徴のダイアグラム４００が例示的に示されている。

【００６４】

ほとんどのプログラムは、個別のプログラムバイナリに格納されるライブラリコードを用いる。この特徴は、関連するプログラムバイナリに対する機能的な依存性を示している

。プログラムは、他のどのバイナリが、より正確に言えば、どの機能を見つけて、適切に動作するように結びつけるべきかを記述するバイナリセクションを有する。

【００６５】

例えば、Ｌｉｎｕｘプラットフォームで普及しているＥＬＦ（Executable and Linkable Format）バイナリフォーマットにおいて、この情報を収集するために、インポートテーブル及びＧＯＴ（Global Offset Table）を用いることができる。他のバイナリフォーマットは、類似のバイナリセクションまたはテーブルを有する。

【００６６】

典型的な一実施形態において、プログラムＡ　４０２は、プログラムＢ　４０４の関数Ｂ１　４０５とプログラムＣ　４０６の関数Ｃ１　４０７とを用いる。この典型的なプログラムからの参照特徴は、ブロック４１０において、より詳細に示される。

【００６７】

再び図３を参照すると、典型的なリソース特徴３１８が以下の表１で示されている。

【００６８】

表１：リソース特徴

【表１】

リソース特徴

種類	値
文字列	「バージョン 1.0.1」
文字列	使用:プログラム
シンボル	関数Ａ
グローバルデータ	192.168.1.0
グローバルデータ	255
アイコン	バイナリデータ

【００６９】

プログラムは、プログラムバイナリに埋め込まれた様々なリソースを使用する。これらのリソースのうちのいくつかは、プログラムで用いられるデータである。例えば、グローバルデータ、プログラムメタデータ、プログラムアイコン、文字列（strings）、デバッグシンボル等は、リソース特徴３１８のこのカテゴリに属する。そのような情報は、通常、個別のバイナリセクションで保存される。例えば、ＥＬＦバイナリフォーマットにおいて、読み出し専用データセクション及びシンボルテーブルセクションがそのような情報のために用いられる。

【００７０】

表１で示したように、リソース特徴のこの例において、このテーブルは、種類（Kind）の列を有し、データは実際に測定及び／または受信した値（Value）である。この例は、いくつかの文字列と、プログラム関数シンボルと、読み出し専用セクションからのグローバルデータと、メタデータに属するアイコンデータとを示している。

【００７１】

次に図５を参照すると、図５には本原理による抽象化制御フロー特徴の生成及び抽出のための方法５００が例示的に示されている。

【００７２】

一実施形態において、与えられたプログラムが、ブロック５０２において、逆アセンブルされ、アルゴリズムは各命令を反復する。ブロック５０４において、命令が制御依存命令（例えば、算術命令）であると判定されない場合、ブロック５０８において、この命令

が廃棄される。本原理の様々な実施形態によれば、以下の典型的な擬似コード１で示すように、ブロック５０４において、命令が制御依存命令であると判定された場合、オペコードのみを取り出し、抽象化制御フロー特徴に含める。

【００７３】

擬似コード１：抽象化制御フロー特徴の生成

【表２】

```
4028b0: mov   $0x413c98,%edi
4028b5: callq 402450
4028ea: xor   %eax,%eax
402905: mov   $0x413800,%edi
40290a: callq 402340
402975: je    405203
402994: test  %eax,%eax
4029ad: jmp   4029e5
402bc4: mov   $0x0,0x218582(%rip)
4048d6: retq
```

【００７４】

制御フロー情報（例えば、ファンクションコール、リターン、ジャンプ及びシステムコール）は、それらの動作を表す重要な記述である。しかしながら、それらの十分な情報（full information）を用いると、特定の細部（certain details）がマイナーチェンジのみで敏感に変化し、ノイズが過度に多くなることがある。例えば、プログラム命令は、他のサブルーチンに対する複数のジャンプ命令を用いるため、バイナリにおけるそれらの位置は、小さなコードパッチで変化する対象（subject）となる。したがって、本原理の様々な実施形態によれば、制御フロー情報のサブセットを用いる。このため、十分な情報を用いるよりも、敏感な変化に対してより回復力が持つ。様々な実施形態によれば、オペコードを含むが、制御依存命令のための命令パラメータを有しない命令情報のサブセット（例えば、ジャンプ、コール及びリターン命令）を用いてもよい。上述した実施形態では、非制御依存命令を用いていない。

【００７５】

再び図３を参照すると、典型的な構造特徴３２２が以下の表２で示される。

【００７６】

表２：構造特徴

【表３】

セクション名	セクションサイズ
.gnu.hash	104
.dynsym	3096
.init	26
.plt	1808
.text	63066
.fini	9
.got.plt	920
.data	596
.bss	3424
.rodata	20732

【００７７】
バイナリの他の特徴は、バイナリの構造情報である。本原理によれば、バイナリセクションの特性（例えば、名称、サイズ及びバイナリセクションの数）が、構造特徴３２２の１つとして用いられる。表２で詳細に記載したように、構造特徴３２２の典型的なテーブルにおいて、１つの列はバイナリセクションの名称を示し、他の列はバイナリセクションのサイズを示している。
【００７８】
次に図６を参照すると、図６には本原理によるハイブリッド特徴類似度比較のためのシステム及び方法６００が例示的に示されている。
【００７９】
様々な実施形態によれば、（例えば、マルウェア攻撃を検出及び／または防止するための）バイナリの複数の特性の類似度を比較及び判定するために、本原理によるプログラムバイナリのハイブリッド特徴を用いてもよい。ブロック６０２、６０４及び６０６において、複数の（例えばＮ個の）プログラムバイナリから生成される１組のＮ個のハイブリッド特徴と、１つまたは複数のハイブリッド特徴パラメータ６０８（例えば、比較における各特徴の貢献を判定するレートのセット）とが、ブロック６１０におけるハイブリッド特徴類似度比較のために入力される。
【００８０】
ハイブリッド特徴パラメータ６０８は、以下のように表すことができる。
Ｃ＿Ｆ：参照特徴のためのパラメータ
Ｃ＿Ｒ：リソース特徴のためのパラメータ
Ｃ＿Ｃ：抽象化制御フロー特徴のためのパラメータ
Ｃ＿Ｓ：構造特徴のためのパラメータ
Ｃ＿Ｆ、Ｃ＿Ｒ、Ｃ＿Ｃ、Ｃ＿Ｓは、０と１との間の比率である。
Ｃ＿Ｆ＋Ｃ＿Ｒ＋Ｃ＿Ｃ＋Ｃ＿Ｓ＝１
Ｃ＿Ｆ＿Ｐ：類似したプログラム参照と照合するための閾値
Ｃ＿Ｆ＿Ｆ：類似した関数参照と照合するための閾値
【００８１】
一実施形態において、組み合わせ生成器６１２は、想定される全てのバイナリ対の組み合わせを生成するように構成されている。ブロック６１６において、各２つのバイナリ（例えば、対）の類似度比較が実行され、ブロック６１８において、ハイブリット差異スコ

アを生成する。本原理によれば、ブロック６１４において、類似度比較（例えば、特徴比較）が１つまたは複数のバイナリ対について反復され、ブロック６２０において、例えばマルウェア攻撃の検出及び防止で用いるために、類似度ベクトルが生成及び出力される。

【００８２】

本原理によれば、ブロック６１０におけるハイブリッド特徴類似度比較は以下のように実行すればよい。

【数６】

```
Hybrid_Feature_Similarity_Comparison(HFList, HFIndex, C_F, C_R, C_C, C_S, C_F_P, C_F_F)

        Explanation of input parameters:

                (1) A list of hybrid features of binary P1, P2, ... , PN

                        HFList = [HF(P1), HF(P2), ... , HF(PN)]

                (2) Index of binaries

                        HFIndex = [P1, P2, ... , PN]

                (3) Hybrid Feature Parameters C_F, C_R, C_C, C_S, C_F_P, C_F_F
```

【００８３】

類似度ベクトル６２０：

【数７】

```
Similarity Vector (Component 620)

                = an empty N * N vector having N rows and N columns
```

【００８４】

組み合わせ生成器６１２：

【数８】

```
        // Make N * N combinations of P1, P2, ... , PN excluding the comparison

        to itself and duplicates

        CombinationList = GenerateCombination(HFIndex)
```

【００８５】

特徴比較６１４の反復：

【数９】

```
        For each (PX, PY) in CombinationList:

                // (Component 616, 618), Note: Definition of Component 618 follows.

                HybridDiffScore (component 618) = HDS(HF(PX), HF(PY), C_F,

        C_R, C_C, C_S, C_F_P, C_F_F)

                Similarity Vector [PX, PY] = HybridDiffScore

        return Similarrity Vector (Component 620)
```

【００８６】

上述したように、ハイブリッド差異スコア６１８は２つのバイナリ間の類似度を表し、全ての組み合わせのスコアは類似度ベクトル６２０にて保存される。いくつかの実施形態

において、本原理によれば、バイナリのクラスタは、例えばクラスタリングアルゴリズムを、類似度ベクトルで保存されたデータに適用することで生成される。

【００８７】

次に図７を参照すると、図７には本原理による２つのバイナリの類似度比較のための方法７００が例示的に示されている。

【００８８】

一実施形態において、本原理によれば、ブロック７２２において、２つのハイブリッドバイナリ特徴７０２、７１２の類似度比較が実行される。比較は、ブロック７２４、７２６、７２８及び７３０において同じ種類の特徴間の差異スコア値を決定するために、同じ種類の２つの特徴（例えば、参照特徴７０４、７１４；リソース特徴７０６、７１６；抽象化制御フロー特徴７０８、７１８；及び構造特徴７１０、７２０）間でそれぞれ実行される。

【００８９】

本原理の様々な実施形態によれば、差異スコア値７２４、７２６、７２８及び７３０は、例えばハイブリッド特徴パラメータ７３２の値を差異スコア値７２４、７２６、７２８及び７３０と乗算することで、ブロック７３４にて決定ハイブリッド差異スコア（determined hybrid difference score）を決定するために用いられる。

【００９０】

一実施形態において、プログラムバイナリＰ１及びＰ２間のハイブリッド差異スコア７３４は以下のように決定すればよい。

【数１０】

$$\begin{aligned}&HDS(HF(P1), HF(P2), C\_F, C\_R, C\_C, C\_S, C\_F\_P, C\_F\_F) =\\&C\_F * Sim\_F(F\_F(P1), F\_F(P2), C\_F\_P, C\_F\_F) + C\_R * Sim\_R(F\_R(P1),\\&F\_R(P2)) + C\_C * Sim\_C(F\_C(P1), F\_C(P2)) + C\_S * Sim\_S(F\_S(P1),\\&F\_S(P2))\end{aligned}$$

【００９１】

Ｓｉｍ＿Ｆ、Ｓｉｍ＿Ｒ、Ｓｉｍ＿Ｃ及びＳｉｍ＿Ｓの定義は以下で説明する。

【００９２】

一実施形態において、プログラムＰ１、Ｐ２の参照特徴７０４、７１４の類似度比較７２４は、以下のように実行すればよい。

【数１１】

```
Sim_F(F_F(P1), F_F(P2), C_F_P, C_F_F)
        Total = Min(|F_F(P1)|, |F_F(P2)|)
        Count = 0
        For (RP1, RF1) from F_F(P1)
          For (RP2, RF2) from F_F(P2)
            If (Difference(RP1, RP2) <= C_F_P and Difference(RF1,
            RF2) <= C_F_F)
                Count+= 1
        return Count / Total
```

ここで、ＲＰはプログラム基準を表し、ＲＦは関数基準を表す。

【００９３】

一実施形態において、プログラムＰ１、Ｐ２のリソース特徴７０６、７１６の類似度比較７２６は、以下のように実行すればよい。

【００９４】
【数１２】

```
Sim_R(F_R(P1), F_R(P2))
        Total = Min(|F_R(P1)|, |F_R(P2)|)
        Count = 0
        For (K1, V1) from F_R(P1)
            For (K2, V2) from F_R(P2)
                If (K1 = K2 and V1 = V2)
                    Count+= 1
        return Count / Total
```

ここで、Ｋは種類（Kind）を表し、Ｖは値（Value）を表す。

【００９５】
一実施形態において、プログラムＰ１、Ｐ２の抽象化制御フロー特徴７０８、７１８の類似度比較７２８は、以下のように実行すればよい。

【数１３】

```
Sim_C(F_C(P1), F_C(P2))
        Score = |LongestCommonSubsequence(F_C(P1), F_C(P2))| /
        min(|F_C(P1)|, |F_C(P2)|)
        return Score
```

【００９６】
一実施形態において、プログラムＰ１、Ｐ２の構造特徴７１０、７２０の類似度比較７３０は、以下のように実行すればよい。

【数１４】

$$Sim\_S(F\_S(P1), F\_S(P2))$$

Scores = []

For (Sec_1, size_1) from F_S(P1)

For (Sec_2, size_2) from F_S(P2)

if (Sec_1 = Sec_2)

$$Score = |size_1 - size_2| / \max(size\_1, size\_2)$$

Score. push(Score)

Sum = 0, Count = 0

For score from Scores:

Sum+= score

Count+= 1

return Sum / Count

【００９７】

様々な実施形態によれば、参照特徴７２４の比較において、例えば類似するプログラム名及び関数名を照合するために閾値を用いてもよい。例えば、同じライブラリの異なるバージョン（例えば、ＬｉｂＸ＿Ｖ１及びＬｉｂＸ＿Ｖ２）は、閾値を用いて照合できる。対応する２つの閾値は、Ｃ＿Ｆ＿Ｐ及びＣ＿Ｆ＿Ｆとして追加される。

【００９８】

次に図８を参照すると、図８には本原理によるハイブリッドプログラムバイナリ特徴の抽出及び比較のための典型的なシステム８００が例示的に示されている。

【００９９】

システム８００の多くの態様が、例示及び明確さのために単数で記載されているが、システム８００の説明に関して言及される要素は、複数の場合にも同じことが適用可能である。例えば、単一のコントローラ８１６が記載されているが、本原理の趣旨を維持する限り、本原理の教示による２つ以上のコントローラ８１６を用いることができる。さらに、ストレージデバイス８１８は、システム８００に含まれる単なる１つの態様のすぎず、本原理の趣旨を維持する限り、複数の場合にも拡張できることを理解されたい。

【０１００】

本原理の様々な実施形態によれば、本原理は、例えばプログラムのホワイトリスト登録、未知のバイナリの特性の判定、悪意のある機能の尤度判定及びマルウェアクラスタリングのために用いてもよい。

【０１０１】

プログラムのホワイトリスト登録に関して、これはソフトウェアの変形または異なるバージョンを判定することで達成してもよい。ソフトウェア会社及び開発者は、バグ修正、セキュリティ更新及び新たな特徴のために、ソフトウェアの様々なバージョンを作成する。例えば、ある会社が数週間または数ヶ月に１度バイナリを更新する場合、全てのバイナリ情報を企業内で取得すると、数十から数百の異なるバージョンのプログラムが存在する可能性がある。

【０１０２】

本発明は、そのような複数のプログラムの類似度を判定することができる。無害なソフ

トウェアの異なるバージョンを知ることは、これらを悪意のあるソフトウェアとの比較から除外して、マルウェア検出の複雑度を低減するために有用である。

【０１０３】

例えば、１０，０００のバイナリのうち、４，０００のバイナリが既知で無害な５００のバイナリの変形であると判定する。この場合、４，０００のバイナリをホワイトリスト登録した後、マルウェアに関して６，０００のバイナリのみを調べればよい。

【０１０４】

未知のバイナリの特性を判定することに関して、未知のソフトウェアバイナリに遭遇したとき、そのバイナリの特性が何であるか（例えば、そのバイナリがユーティリティプログラムであるか、ワードプロセッサであるか、またはマルウェアであるか）は分からない。本発明を用いて未知のバイナリを既知のソフトウェアのリストと比較することで、該バイナリの特性を正確に判定することができる。例えば、他のソフトウェアとの比較において、以下の類似度を得た場合：

－未知のプログラムとファイル検索ユーティリティとの類似度：２０％、

－未知のプログラムとネットワークユーティリティとの類似度：８０％、

本原理によれば、このプログラムは、ネットワーク機能及びファイル検索機能を有する可能性があると判定される。

【０１０５】

悪意のある機能の尤度を推定することに関して、未知のバイナリを悪意のあるソフトウェアバイナリのリストと比較してもよく、該バイナリが悪意のある機能を有する可能性を判定してもよい。例えば：

－未知のプログラムとマルウェアＸとの類似度：５０％、

－未知のプログラムとファイル検索ユーティリティとの類似度：２０％、

－未知のプログラムとネットワークユーティリティとの類似度：３０％。

このプログラムは、マルウェアＸで見られる悪意のある機能を有する可能性（例えば、尤度百分率）があると判定される。

【０１０６】

マルウェアクラスタリングにおいて、未知のバイナリが複数種類の悪意のあるソフトウェアとの類似度を示すとき、本発明は、ウィルス対策アプリケーションで用いるマルウェアクラスタリングによりマルウェアのカテゴリを理解するために役に立つ。例えば：

－未知のプログラムとマルウェアファミリー１との類似度：７０％

－未知のプログラムとマルウェアファミリー２との類似度：２７％

－未知のプログラムとマルウェアファミリー３との類似度：３％

本原理によれば、このバイナリは、マルウェアファミリー２または３よりもマルウェアファミリー１により密接に関連していると判定され、それに応じて、ウィルス対策／マルウェア対策アプリケーションが悪意のあるソフトウェア攻撃を検出及び防止するために更新／適用される。

【０１０７】

システム８００は、１つまたは複数のコンピューティングネットワーク及び／またはストレージデバイス８１８と接続可能なバス８０１を含むことができる。バイナリ特徴の抽出のためにプログラムバイナリ特徴抽出器８０２を用いてもよく、ハイブリッドバイナリ特徴生成器８０４を用いてハイブリッド特徴を生成してもよい。ハイブリッドバイナリ特徴は、例えばハイブリッド特徴パラメータ判定装置８０８によって提供されるハイブリッド特徴パラメータを入力としてさらに取得できる、ハイブリッドバイナリ特徴類似度解析器８０６を用いて解析してもよい。

【０１０８】

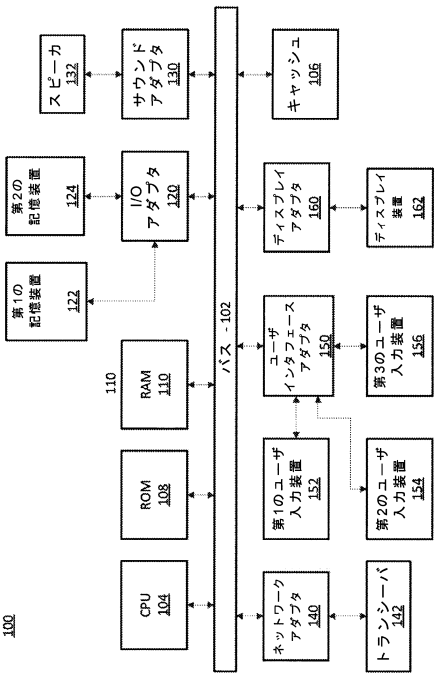
様々な実施形態によれば、バイナリ対どうしの差異を判定するために類似度判定装置８１２を用いてもよく、類似度ベクトル生成器８１０は、類似度比較に基づいて、類似度スコアベクトルを生成して出力する。生成された類似度ベクトルは、マルウェア攻撃解析器、検出器及びプリベンタ８１４において、マルウェア攻撃保護のために（例えば、リアル

タイムで、将来において）用いてもよい。それは、例えば類似度ベクトルに基づいてマルウェア定義を更新すること、更新されたマルウェア定義によって検出されたマルウェアを隔離すること等を（例えば、手動または自動で）ウィルス対策ソフトウェアに命令するためにコントローラ８１６によって制御できる。本原理の様々な実施形態によれば、例えば、マルウェア攻撃の検出及び防止に使用するため、更新されたマルウェア定義、類似度比較の結果等を保存するためにストレージデバイス８１８を用いてもよい。

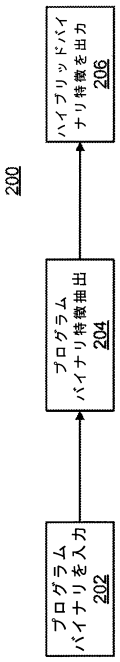
【０１０９】

上記は、あらゆる観点において例示的（illustrative）かつ典型的（exemplary）であって限定的でないものと理解されるべきであり、本明細書で開示する本発明の範囲は、詳細な説明から決定されるべきではなく、特許法で認められた最大限の広さに基づいて解釈される特許請求の範囲から決定されるべきである。本明細書中に図示及び記載されている実施形態は、本発明の原理を説明するものにすぎず、本発明の範囲及び主旨から逸脱することなく当業者は様々な変更を実施することができることを理解されたい。当業者は、本発明の範囲及び精神から逸脱することなく、様々な他の特徴の組み合わせを実施できる。以上、本発明の態様について、特許法で要求される詳細及び特殊性と共に説明したが、特許証で保護されることを要求する特許請求の範囲は、添付の特許請求の範囲に示されている。

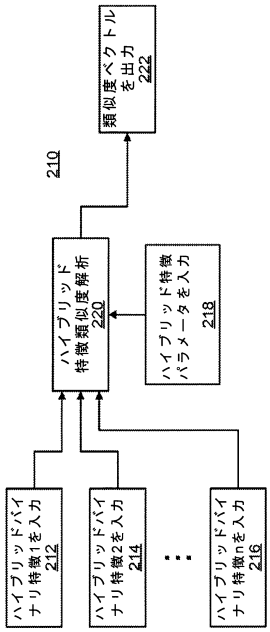
【図１】



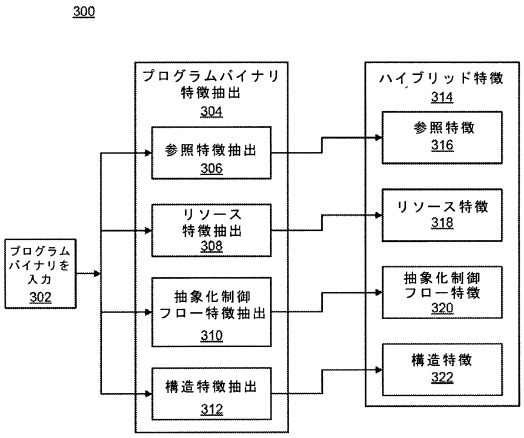
【図２Ａ】



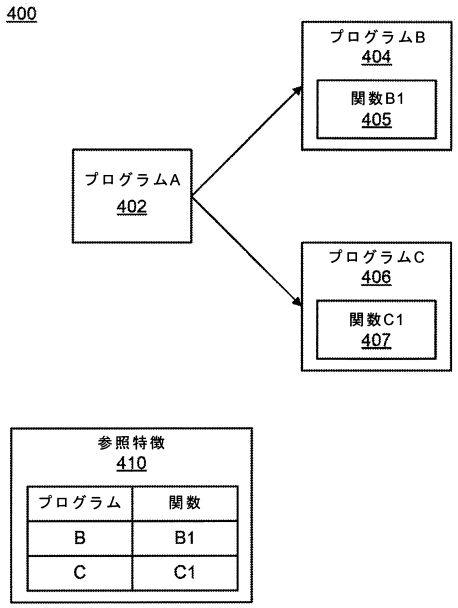
【図２Ｂ】



【図３】



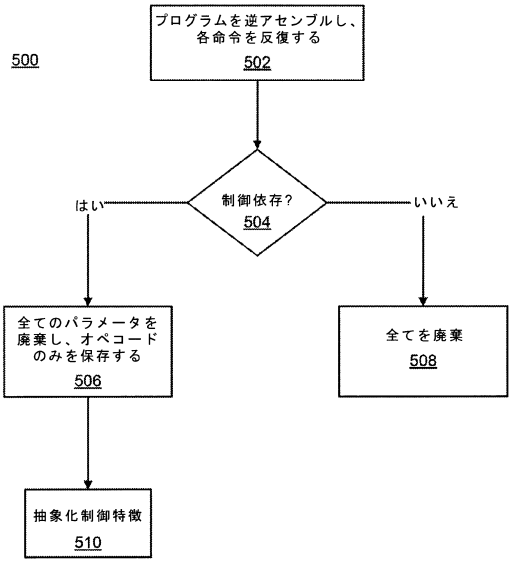
【図４】



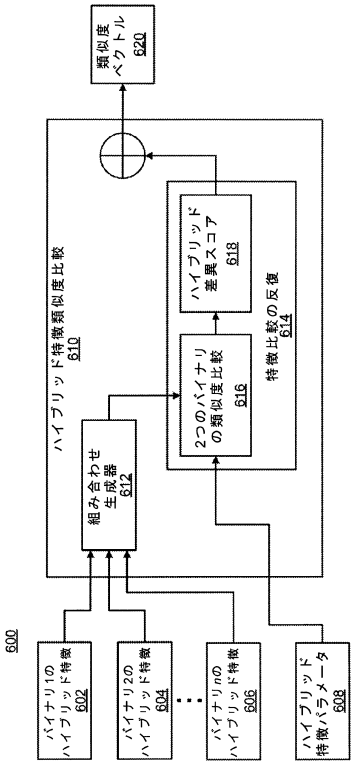
参照特徴
410

プログラム	関数
B	B1
C	C1

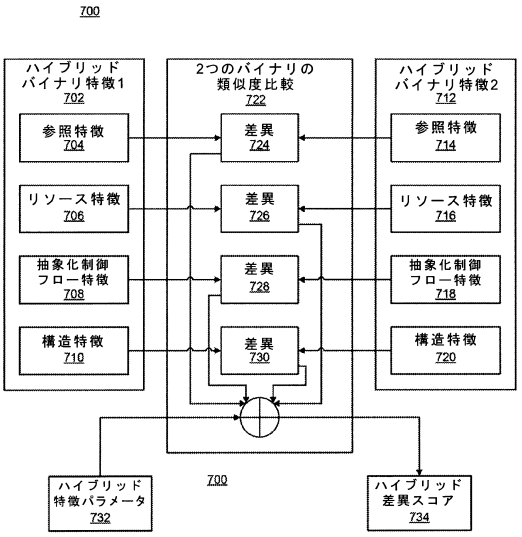
【図５】



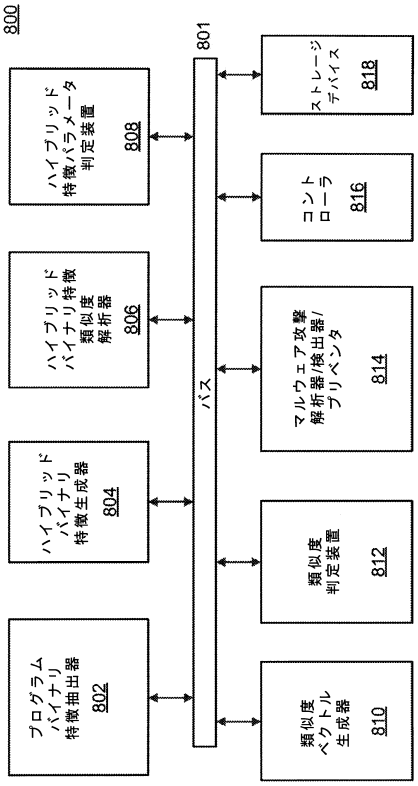
【図６】


600
バイナリ1のハイブリッド特徴 602
バイナリ2のハイブリッド特徴 604
バイナリnのハイブリッド特徴 606
ハイブリッド特徴パラメータ 608
ハイブリッド特徴類似度比較 610
組み合わせ生成器 612
2つのバイナリの類似度比較 616
特徴比較の反復 614
ハイブリッド差異スコア 618
類似度ベクトル 620

【図７】


700
ハイブリッドバイナリ特徴1 702
参照特徴 704
リソース特徴 706
抽象化制御フロー特徴 708
構造特徴 710
2つのバイナリの類似度比較 722
差異 724
差異 726
差異 728
差異 730
ハイブリッドバイナリ特徴2 712
参照特徴 714
リソース特徴 716
抽象化制御フロー特徴 718
構造特徴 720
700
ハイブリッド特徴パラメータ 732
ハイブリッド差異スコア 734

【図８】


800
801
バス
ハイブリッド特徴パラメータ判定装置 808
ハイブリッドバイナリ特徴類似度解析器 806
ハイブリッドバイナリ特徴生成器 804
プログラムバイナリ特徴抽出器 802
ストレージデバイス 818
コントローラ 816
マルウェア攻撃解析器/検出器/プリベンタ 814
類似度判定装置 812
類似度ベクトル生成器 810

フロントページの続き

(72)発明者　リー、　ジュンワン
アメリカ合衆国　０８５４０　ニュージャージー州　プリンストン　クロムウェル　コート　６
(72)発明者　リ、　ジチュン
アメリカ合衆国　０８５４０　ニュージャージー州　プリンストン　セイヤー　ドライヴ　３０６
(72)発明者　ウ、　ジェンユ
アメリカ合衆国　０８５３６　ニュージャージー州　プレインスボロ　マーション　レーン　４
(72)発明者　ジー、　カンクック
アメリカ合衆国　０８５４０　ニュージャージー州　プリンストン　ジョナソン　デイトン　コート　１９２
(72)発明者　ジアン、　グオフェイ
アメリカ合衆国　０８５４０　ニュージャージー州　プリンストン　ダンビー　コート　５

審査官　田名網　忠雄

(56)参考文献　特表２０１４－５３４５３１（ＪＰ，Ａ）
特開２０１３－０７７１５４（ＪＰ，Ａ）
特開２０１２－０２７７１０（ＪＰ，Ａ）
特開２０１０－１９８５６５（ＪＰ，Ａ）
国際公開第２０１５／０９９７８０（WO，Ａ1）
米国特許出願公開第２０１２／０１５９６２５（ＵＳ，Ａ1）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　２１／５０－２１／５７