(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6949951号
(P6949951)

(45)発行日　令和3年10月13日(2021.10.13)　(24)登録日　令和3年9月27日(2021.9.27)

(51)Int.Cl.　F I
G06F 21/52 (2013.01)　G06F 21/52
G06F 21/56 (2013.01)　G06F 21/56　360

請求項の数 38　(全 35 頁)

(21)出願番号　特願2019-518173(P2019-518173)
(86)(22)出願日　平成29年6月16日(2017.6.16)
(65)公表番号　特表2019-519056(P2019-519056A)
(43)公表日　令和1年7月4日(2019.7.4)
(86)国際出願番号　PCT/US2017/037841
(87)国際公開番号　WO2017/218872
(87)国際公開日　平成29年12月21日(2017.12.21)
審査請求日　令和2年5月21日(2020.5.21)
(31)優先権主張番号　62/350,917
(32)優先日　平成28年6月16日(2016.6.16)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　518444820
ヴァーセック　システムズ，インコーポレイテッド
アメリカ合衆国，カリフォルニア州　９５１１０，サン　ノゼ，エアポート　パーク　ウェイ　２２６，スイート　３５０
(74)代理人　100079108
弁理士　稲葉　良幸
(74)代理人　100109346
弁理士　大貫　敏史
(74)代理人　100117189
弁理士　江口　昭彦
(74)代理人　100134120
弁理士　内藤　和彦

最終頁に続く

(54)【発明の名称】コンピュータアプリケーション内のメモリ破損を修復するためのシステム及び方法

(57)【特許請求の範囲】
【請求項１】
コンピュータによって実装される方法であって、
ロード時間中にコンピュータアプリケーションのモデルを抽出すること、
前記コンピュータアプリケーションの前記モデルを記憶すること、
実行時にデータを収集するために前記コンピュータアプリケーション内に命令を挿入すること、
前記コンピュータアプリケーションの前記記憶済みのモデルに対して実行時に収集した前記データを分析して１つ又は複数のセキュリティイベントの検出を行うこと、
前記１つ又は複数のセキュリティイベントを前記検出することに基づいて、
前記コンピュータアプリケーションに関連する少なくとも１つのアクティブプロセス又はスレッドの実行を一時停止すること、及び
前記コンピュータアプリケーションの継続的実行を保つやり方で前記コンピュータアプリケーションに関連する前記少なくとも１つのアクティブプロセス又はスレッドに関連する少なくとも１つのコンピュータルーチンを修正すること、
を含む、方法。
【請求項２】
前記少なくとも１つのコンピュータルーチンが前記少なくとも１つのプロセスに関連して実行される、請求項１に記載の方法。
【請求項３】

前記１つ又は複数の検出されるセキュリティイベントが、前記コンピュータアプリケーション内の別のコードパスへの悪意ある移動に関連する、請求項１に記載の方法。
【請求項４】
修正することが、前記コンピュータアプリケーションに関連するパッチ又は構成を検査することを含む、請求項１に記載の方法。
【請求項５】
１つ又は複数の集約パッチが利用者によって受信されることに応答して、
前記コンピュータアプリケーションに関連する前記少なくとも１つのコンピュータルーチンを修正又は除去すること、及び
前記コンピュータアプリケーションに関連する１つ又は複数の個々のパッチを修正又は除去すること
の少なくとも１つを実行すること
を更に含む、請求項１に記載の方法。
【請求項６】
前記少なくとも１つのコンピュータルーチンに関連する１つ又は複数のスタックを修正すること
を更に含む、請求項１に記載の方法。
【請求項７】
前記少なくとも１つのコンピュータルーチンに関連する１つ又は複数のヒープを修正すること
を更に含む、請求項１に記載の方法。
【請求項８】
前記少なくとも１つのアクティブプロセスが前記少なくとも１つのコンピュータルーチンを実行している間、前記少なくとも１つのアクティブプロセスに関連する前記少なくとも１つのコンピュータルーチンを修正すること
を更に含む、請求項１に記載の方法。
【請求項９】
前記少なくとも１つのコンピュータルーチンを修正した後、前記少なくとも１つのアクティブプロセス又はスレッドの実行を再開すること、
を更に含む、請求項１に記載の方法。
【請求項１０】
コンピュータシステムであって、
ロード時間中にコンピュータアプリケーションのモデルを抽出し、
前記コンピュータアプリケーションの前記モデルを記憶し、
実行時にデータを収集するために前記コンピュータアプリケーション内に命令を挿入する
ように構成されるインストルメンテーションエンジンと、
前記コンピュータアプリケーションの前記記憶済みのモデルに対して実行時に収集した前記データを分析して１つ又は複数のセキュリティイベントの検出を行い、
前記１つ又は複数のセキュリティイベントを前記検出することに基づいて、
前記コンピュータアプリケーションに関連する少なくとも１つのアクティブプロセス又はスレッドの実行を一時停止し、
前記コンピュータアプリケーションの継続的実行を保つやり方で前記コンピュータアプリケーションに関連する前記少なくとも１つのアクティブプロセス又はスレッドに関連する少なくとも１つのコンピュータルーチンを修正する
ように構成される分析エンジンと
を含む、コンピュータシステム。
【請求項１１】
前記少なくとも１つのコンピュータルーチンが前記少なくとも１つのアクティブプロセスに関連して実行される、請求項１０に記載のシステム。

【請求項１２】
前記１つ又は複数の検出されるセキュリティイベントが、前記コンピュータアプリケーション内の別のコードパスへの悪意ある移動に関連する、請求項１０に記載のシステム。
【請求項１３】
前記分析エンジンが、前記コンピュータアプリケーションに関連するパッチ又は構成を検査するように更に構成される、請求項１０に記載のシステム。
【請求項１４】
前記分析エンジンが
１つ又は複数の集約パッチが利用者によって受信されることに応答して、
前記コンピュータアプリケーションに関連する前記少なくとも１つのコンピュータルーチンを修正又は除去すること、及び
前記コンピュータアプリケーションに関連する１つ又は複数の個々のパッチを修正又は除去すること
の少なくとも１つを実行するように更に構成される、請求項１０に記載のシステム。
【請求項１５】
前記分析エンジンが
前記少なくとも１つのコンピュータルーチンに関連する１つ又は複数のスタックを修正する
ように更に構成される、請求項１０に記載のシステム。
【請求項１６】
前記分析エンジンが
前記少なくとも１つのコンピュータルーチンに関連する１つ又は複数のヒープを修正する
ように更に構成される、請求項１０に記載のシステム。
【請求項１７】
前記分析エンジンが
前記少なくとも１つのアクティブプロセスが前記少なくとも１つのコンピュータルーチンを実行している間、前記少なくとも１つのアクティブプロセスに関連する前記少なくとも１つのコンピュータルーチンを修正する
ように更に構成される、請求項１０に記載のシステム。
【請求項１８】
前記分析エンジンが
前記少なくとも１つのコンピュータルーチンを修正した後、前記少なくとも１つのアクティブプロセス又はスレッドの実行を再開する
ように更に構成される、請求項１０に記載のシステム。
【請求項１９】
コンピュータによって実装される方法であって、
ロード時間中にコンピュータアプリケーションのモデルを抽出すること、
前記コンピュータアプリケーションの前記モデルを記憶すること、
実行時にデータを収集するために前記コンピュータアプリケーション内に命令を挿入すること、
前記コンピュータアプリケーションの前記記憶済みのモデルに対して実行時に収集した前記データを分析して１つ又は複数のセキュリティイベントの検出を行うこと、
前記１つ又は複数のセキュリティイベントを前記検出すると、１つ又は複数のリターン命令を実行する前に前記コンピュータアプリケーションに関連するメモリ破損を一時的に修復すること、
前記１つ又は複数の検出済みのセキュリティイベントに基づいて実用的な情報を報告すること、及び
前記１つ又は複数のセキュリティイベントを前記検出することに基づいて、
前記コンピュータアプリケーションに関連する少なくとも１つのアクティブプロセス又

はスレッドの実行を一時停止すること、及び
前記コンピュータアプリケーションの継続的実行を保つやり方で前記コンピュータアプリケーションに関連する前記少なくとも１つのアクティブプロセス又はスレッドに関連する少なくとも１つのコンピュータルーチンを修正すること
を含む、方法。
【請求項２０】
少なくとも１つのプロセスが実行している間に前記少なくとも１つのプロセスに関連する少なくとも１つのコンピュータ命令を修正すること
を更に含む、請求項１９に記載の方法。
【請求項２１】
前記少なくとも１つのコンピュータルーチンを修正した後、前記少なくとも１つのアクティブプロセス又はスレッドの実行を再開すること
を更に含む、請求項１９に記載の方法。
【請求項２２】
コンピュータシステムであって、
ロード時間中にコンピュータアプリケーションのモデルを抽出し、
前記コンピュータアプリケーションの前記モデルを記憶し、
実行時にデータを収集するために前記コンピュータアプリケーション内に命令を挿入する
ように構成されるインストルメンテーションエンジンと、
前記コンピュータアプリケーションの前記記憶済みのモデルに対して実行時に収集した前記データを分析して１つ又は複数のセキュリティイベントの検出を行い、
前記１つ又は複数のセキュリティイベントを前記検出すると、１つ又は複数のリターン命令を実行する前に前記コンピュータアプリケーションに関連するメモリ破損を一時的に修復し、
前記１つ又は複数の検出済みのセキュリティイベントに基づいて実用的な情報を報告し、
前記１つ又は複数のセキュリティイベントを前記検出することに基づいて、
前記コンピュータアプリケーションに関連する少なくとも１つのアクティブプロセス又はスレッドの実行を一時停止し、
前記コンピュータアプリケーションの継続的実行を保つやり方で前記コンピュータアプリケーションに関連する前記少なくとも１つのアクティブプロセス又はスレッドに関連する少なくとも１つのコンピュータルーチンを修正する
ように構成される分析エンジンと
を含む、コンピュータシステム。
【請求項２３】
前記分析エンジンが
少なくとも１つのプロセスが実行している間に前記少なくとも１つのプロセスに関連する少なくとも１つのコンピュータ命令を修正する
ように更に構成される、請求項２２に記載のシステム。
【請求項２４】
前記分析エンジンが
前記少なくとも１つのコンピュータルーチンを修正した後、前記少なくとも１つのアクティブプロセス又はスレッドの実行を再開する
ように更に構成される、請求項２２に記載のシステム。
【請求項２５】
コンピュータアプリケーションの１つ又は複数のコードの脆弱性について、それぞれのコードの脆弱性をメモリ内のテーブル内の個々のシステム応答にマップすること、
前記コンピュータアプリケーションのコードの脆弱性にアクセスするイベントを検出すること、

前記イベントを前記検出することに応答して、前記アクセスされるコードの脆弱性に前記メモリ内のテーブル内でマップされるシステム応答を動的に決定すること、及び
前記決定したシステム応答を実行することであって、前記実行することは前記アクセスされるコードの脆弱性を前記イベントが利用するのを防ぐ、実行すること
を含む、コンピュータによって実装される方法。
【請求項２６】
少なくとも１つのコードの脆弱性及びマップされたシステム応答が前記コンピュータアプリケーションの開発者から提供される、請求項２５に記載の方法。
【請求項２７】
少なくとも１つのコードの脆弱性及びマップされたシステム応答が前記コンピュータアプリケーションのロード時に又は実行時にコードアナライザによって自動で決定される、請求項２５に記載の方法。
【請求項２８】
前記システム応答が、システム又は利用者によってプログラム可能なシステムコールバックルーチンを含む、請求項２５に記載の方法。
【請求項２９】
カーネルモード例外ハンドラを上書きするために例外ハンドラを計装すること、
前記コードの脆弱性の前記アクセスに応答して例外をトリガすること、
前記計装された例外ハンドラにおいて前記トリガされる例外を傍受し、前記コードの脆弱性を関連付けること、
前記メモリ内のテーブル内の前記関連するコードの脆弱性を前記計装された例外ハンドラによって照会することであって、前記照会することは前記コードの脆弱性にマップされる前記システムコールバックルーチンを返す、照会すること、及び
前記コードの脆弱性を前記イベントが利用するのを防ぐための命令を開始するために、前記システムコールバックルーチンを前記計装された例外ハンドラによって実行すること
を更に含む、請求項２８に記載の方法。
【請求項３０】
前記システム応答が、
システムログ内に誤りとして前記コードの脆弱性の前記アクセスのログをとること、
前記アクセスされたコードの脆弱性を含むアプリケーションプロセスのイメージをダンプすること、
前記コードの脆弱性の前記アクセス前の前記コンピュータアプリケーションのコピーを復元すること、
メモリから１つ又は複数の修復パッチを動的にロードすることであって、前記ロードすることは前記コードの脆弱性を含む少なくとも１つのコンピュータルーチンを修正するために前記修復パッチを使用し、前記コンピュータアプリケーションを再開することなしに前記少なくとも１つのコンピュータルーチンを修正する、動的にロードすること、
前記アクセスされたコードの脆弱性に基づいて終了が生じるまで前記コンピュータアプリケーションを実行し続けること、及び
前記コンピュータアプリケーションを先回りして終了すること
のうちの１つ又は複数を含む、請求項２５に記載の方法。
【請求項３１】
動的にロードすることが、前記コンピュータアプリケーションを実行しているサーバのメモリから前記修復パッチを直接注入することを含む、請求項３０に記載の方法。
【請求項３２】
コンピュータシステムであって、
コンピュータアプリケーションの１つ又は複数のコードの脆弱性について、それぞれのコードの脆弱性をメモリ内のテーブル内の個々のシステム応答にマップする
ように構成されるインストルメンテーションエンジンと、
前記コンピュータアプリケーションのコードの脆弱性にアクセスするイベントを検出す

ること、
前記イベントを前記検出することに応答して、前記アクセスされるコードの脆弱性に前記メモリ内のテーブル内でマップされるシステム応答を動的に決定すること、及び
前記決定したシステム応答を実行することであって、前記実行することは前記アクセスされるコードの脆弱性を前記イベントが利用するのを防ぐ、実行すること
を行うように構成される分析エンジンと
を含む、コンピュータシステム。
【請求項３３】
少なくとも１つのコードの脆弱性及びマップされたシステム応答が前記コンピュータアプリケーションの開発者から提供される、請求項３２に記載のシステム。
【請求項３４】
少なくとも１つのコードの脆弱性及びマップされたシステム応答が前記コンピュータアプリケーションのロード時に又は実行時にコードアナライザによって自動で決定される、請求項３２に記載のシステム。
【請求項３５】
前記システム応答が、システム又は利用者によってプログラム可能なシステムコールバックルーチンを含む、請求項３２に記載のシステム。
【請求項３６】
前記分析エンジンが、
カーネルモード例外ハンドラを上書きするために例外ハンドラを計装すること、
前記コードの脆弱性の前記アクセスに応答して例外をトリガすること、
前記計装された例外ハンドラにおいて前記トリガされる例外を傍受し、前記コードの脆弱性を関連付けること、
前記テーブル内の前記関連するコードの脆弱性を前記計装された例外ハンドラによって照会することであって、前記照会することは前記コードの脆弱性にマップされる前記システムコールバックルーチンを返す、照会すること、及び
前記コードの脆弱性を前記イベントが利用するのを防ぐための命令を開始するために、前記システムコールバックルーチンを前記計装された例外ハンドラによって実行すること
を行うように更に構成される、請求項３５に記載のシステム。
【請求項３７】
前記システム応答が、
システムログ内に誤りとして前記コードの脆弱性の前記アクセスのログをとること、
前記アクセスされたコードの脆弱性を含むアプリケーションプロセスのイメージをダンプすること、
前記コードの脆弱性の前記アクセス前の前記コンピュータアプリケーションのコピーを復元すること、
メモリから１つ又は複数の修復パッチを動的にロードすることであって、前記ロードすることは前記コードの脆弱性を含む少なくとも１つのコンピュータルーチンを修正するために前記修復パッチを使用し、前記コンピュータアプリケーションを再開することなしに前記少なくとも１つのコンピュータルーチンを修正する、動的にロードすること、
前記アクセスされたコードの脆弱性に基づいて終了が生じるまで前記コンピュータアプリケーションを実行し続けること、及び
前記コンピュータアプリケーションを先回りして終了すること
のうちの１つ又は複数を含む、請求項３２に記載のシステム。
【請求項３８】
前記動的にロードすることが、前記コンピュータアプリケーションを実行しているサーバのメモリから、前記修復パッチを直接注入することを含む、請求項３７に記載のシステム。
【発明の詳細な説明】
【技術分野】

【０００１】
関連出願の相互参照
本願は２０１６年６月１６日に出願された米国仮特許出願第６２／３５０，９１７号の利益を主張する。上記の出願の全教示を参照により全て本明細書に援用する。
【背景技術】
【０００２】
背景
ネットワークにアクセス可能なアプリケーションは、悪意ある攻撃者によって遠隔的に引き起こされるメモリ破損攻撃に対して脆弱であることが多い。リモートユーザのコンピュータネットワークへの新たなアクセスを多くの場合高い特権と共に得ることができるので、悪意ある攻撃者はそのような脆弱性を懸命に利用しようとしている。制御を奪うと、攻撃者はあたかもリモートユーザが危険にさらされたマシンを所有するかのように攻撃者の選択の任意のコードを実行することができる。通常、悪意ある攻撃者の目的は利用者から個人情報及び／又は機密情報を抽出することだが、目的は生産性の喪失を負わせるために利用者の個人活動又は事業活動を混乱させることも含み得る。
【０００３】
予備攻撃は、アプリケーションのメモリアドレス空間内のスタック、ヒープセグメント、並びにインポート、エクスポート、仮想ポインタ（ＶＰＴＲ）、及びシステムコール／システムディスパッチテーブルを含む他のジャンプテーブルに対するバッファ内に戦略的データを配置することによってその準備を助けることができる。予備攻撃は、本来備わっているアプリケーションの一部であるコードの代わりに悪意あるハッカーによって設計されたコードを実行させる最終目的のために、後で仕掛けられる攻撃が実行フローを操作することを可能にする。最も洗練された攻撃者は標的アプリケーションのメモリ空間内に自らの悪意あるコードを直接挿入する必要さえなく、代わりに攻撃者は正当にロードされたアプリケーションコードから選択的に選んだ（即ち望ましいものだけを選んだ）コードのチャンクをまとめることによって既存のコードに再度目的をもたせ、それにより自らの非道な目的を実行することができる。このような高度な実行時のメモリ破損攻撃からアプリケーションを実行時に保護する差し迫った需要がある。
【発明の概要】
【課題を解決するための手段】
【０００４】
概要
本開示の実施形態は、１つ又は複数の実行中プロセス内のメモリ破損によって促進される悪意ある攻撃を防ぐためのシステム及び方法の例を対象とする。一部の実施形態では、システムが悪意ある攻撃を防ぐ操作を実行するための１つ又は複数のインストルメンテーションエンジンと１つ又は複数の分析エンジンとを含む。１つ又は複数のインストルメンテーションエンジンは、１つ又は複数の分析エンジンと同じ又は異なるハードウェア又はコンピュータシステム上に位置し得る。一部の実施形態では、このシステム及び方法は、アプリケーションコードがメモリ内に最初にロードするときコンピュータアプリケーションのモデルを抽出することができる。モデルは、これだけに限定されないが、正当な出所メモリアドレス及び宛先メモリアドレスの対の構築、遷移、基本ブロック境界情報、コードセグメント境界、インポート及びエクスポートアドレステーブル境界、ジャンプテーブル境界、又は当業者に知られている他の任意の種類のコンピュータルーチン関連情報を含み得る。一部の実施形態では、システム及び方法がコンピュータアプリケーションのモデルを記憶し得る。
【０００５】
一部の実施形態では、システム及び方法は、実行時のデータ及び／又はアプリケーションの実行状態を収集するために、コンピュータアプリケーション命令がメモリ内で実行される前に（任意選択的に実行時に）コンピュータアプリケーション内に命令を挿入することができる。一部の実施形態では、システム及び方法がコンピュータアプリケーションの

記憶済みのモデルに対して実行時に収集したデータを分析して１つ又は複数のセキュリティイベントの検出を行うことができる。一部の実施形態では、システム及び方法が１つ又は複数のセキュリティイベントの検出に基づいて、コンピュータアプリケーションの継続的実行を保つやり方でコンピュータアプリケーションに関連する少なくとも１つのアクティブプロセスに関連する少なくとも１つのコンピュータルーチンを修正する（即ちパッチを挿入する）ことができる。

【０００６】

一部の実施形態によれば、コンピュータルーチンが少なくとも１つのプロセスに関連して実行され得る。一部の実施形態では、１つ又は複数の検出されるセキュリティイベントが、コンピュータアプリケーション内の別の（異常な）コードパスへの悪意ある移動に関連し得る。かかる悪意ある移動は、これだけに限定されないが、悪意あるジャンプルーチン、悪意あるコードへのトランポリン、間接ジャンプベクタ、又は当業者に知られている他の任意の悪意ある移動を含み得る。

【０００７】

１つ又は複数の集約パッチが利用者によって受信されることに応答して、一部の実施形態はコンピュータアプリケーションに関連する少なくとも１つのコンピュータルーチンを修正又は除去する、及びコンピュータアプリケーションに関連する１つ又は複数の個々のパッチを修正又は除去する少なくとも１つの操作を実行することができる。一部の実施形態によれば、修正することはコンピュータアプリケーションに関連するパッチ又は構成を検査することを含み得る。一部の実施形態によれば、システム及び方法が少なくとも１つのコンピュータルーチンに関連するスタックを修正することができる。一部の実施形態では、システム及び方法が少なくとも１つの実行中のコンピュータルーチンに関連する１つ又は複数のヒープを修正することができる。他の一部の実施形態では、システム及び方法が１つ又は複数のジャンプテーブルを修正することができる。

【０００８】

更に一部の実施形態では、少なくとも１つのアクティブプロセスが少なくとも１つのコンピュータルーチンを実行している間、システム及び方法が少なくとも１つのプロセスに関連する少なくとも１つのコンピュータルーチンを修正することができる。そのため、一部の実施形態はホットパッチング（又はライブパッチング若しくは動的ソフトウェアパッチング／更新）を使用することができる。一部の実施形態によれば、ホットパッチの結果として置換関数（即ち別の関数）を呼び出すことができる。一部の実施形態では、システム及び方法が、少なくとも１つのコンピュータルーチンを修正する前に少なくとも１つのアクティブプロセス（又はコンピュータアプリケーション）の実行を一時停止することができる。一部の実施形態では、少なくとも１つのコンピュータ命令を修正した後、システム及び方法が少なくとも１つのアクティブプロセスの実行を再開することができる。

【０００９】

一部の実施形態では、システム及び方法が、ロード時間中にコンピュータアプリケーションのモデルを抽出することができる。一部の実施形態によれば、システム及び方法はコンピュータアプリケーションのモデルを記憶し得る。一部の実施形態では、システム及び方法は、実行時にデータを収集するために、コンピュータアプリケーションがメモリ内で実行される前に（任意選択的にメモリ内の）コンピュータアプリケーション内に命令を挿入することができる。一部の実施形態では、システム及び方法がコンピュータアプリケーションの記憶済みのモデルに対して実行時に収集したデータを分析して１つ又は複数のセキュリティイベントの検出を行うことができる。一部の実施形態では、システム及び方法は１つ又は複数のセキュリティイベントを検出すると、１つ又は複数のリターン命令を実行する前にコンピュータアプリケーションに関連するメモリ破損を一時的に修復する（即ち１つ又は複数のポインタを復元する）ことができる。一部の実施形態では、システム及び方法が１つ又は複数の検出済みのセキュリティイベントに基づいて実用的な情報を報告する（即ちパッチを作成するためにベンダに情報を報告する）ことができる。実用的な情報は、これだけに限定されないが、セキュリティイベントが起こる場所／方法、トランポ

リンが行われる場所／方法、脆弱な関数のスタック又はヒープ内でメモリが破損させられた場所／方法等の情報を含み得る。

【００１０】

一部の実施形態では、システム及び方法が、コンピュータアプリケーションの継続的実行を保つやり方で少なくとも１つのプロセスに関連する少なくとも１つのコンピュータルーチンを修正することができる。一部の実施形態では、ベンダからのライトパッチ又はフルパッチが利用者によって受信されると、アプリケーション実行時モニタリング及び分析（ＡＲＭＡＳ）アプリケーションを無効にすることができ、プロセスをシャットダウンすることなしにプロセスに関連するコンピュータメモリ（即ちメモリ内の別の位置）内に新たなコードをロードすることができる。一部の実施形態は、共有された静的ライブラリ又は動的ライブラリとしてベンダによってリリースされるかかるライトパッチを導入することができる。更に一部の実施形態では、システム及び方法が少なくとも１つのプロセスに関連する少なくとも１つのコンピュータ命令を少なくとも１つのプロセスの実行中に修正することができる。一部の実施形態では、システム及び方法が、少なくとも１つのコンピュータ命令を修正する前にコンピュータアプリケーションに関連する少なくとも１つのプロセスの実行を一時停止することができる。一部の実施形態では、少なくとも１つのコンピュータ命令を修正した後、システム及び方法が少なくとも１つのプロセスの実行を再開することができる。

【００１１】

一部の実施形態は、パッチが導入されるまで悪意ある攻撃をリアルタイムで修復する又は防ぐことができる。一部の実施形態は、ソフトウェアベンダの開発者に実用的な修復パスを提供することができる。一部の実施形態は、ヒープベースのコード及び／又はスタックベースのコードのトランポリンをリアルタイムで検出することができる。一部の実施形態は、プロセスを終了することなしにライトパッチをホット導入する（即ちホットパッチングする）ことができる。言い換えれば、一部の実施形態はライトパッチをダウンロードし検査し、プロセス内の全てのスレッドの実行を一時的に停止し、バイナリをホットパッチし、プロセス（及び／又はアプリケーション）内の全てのスレッドを最終的に再開することができる。一部の実施形態は、どのパッチがどの親バイナリ内のどのルーチンと結び付くのかの状態を結び付け（即ち関連付け）、追跡することができる。かかる状態は、これだけに限定されないがライトパッチに関連するチェックサム及び元のバイナリ自体の中のアドレスを含み得る。一部の実施形態は、プロセスに現在関連している状態の結び付きをパッチの導入の前又は後で解消する（即ち関連付けを解く）ことができる。一部の実施形態は状態の関係をハッカーから守ることができる。１つ又は複数の状態を修正すること（又はパッチからオン又は複数の状態を関連付けること若しくはその関連付けを解くこと）により、一部の実施形態は状態の関係をハッカーから守ることができる。一部の実施形態は、これだけに限定されないが（ａ）パッチに関連する（又は含まれる）チェックサムの検査、（ｂ）削除される元のパッチの別のコピーを得ること、及び／又は（ｃ）パッチのコンテンツを暗号化することを含む保護を含み、それにより介入者（ＭＩＭ）攻撃並びに故意の／不注意の削除を防ぐ。

【００１２】

例示する実施形態では、コンピュータアプリケーションの１つ又は複数のコードの脆弱性について、システム及び方法がコードの脆弱性のそれぞれを（例えばセキュリティポリシデータベース内の）メモリ内のテーブル内の個々のシステム応答にマップする。一部の実施形態例では、少なくとも１つのコードの脆弱性及びマップされたシステム応答がコンピュータアプリケーションの開発者から提供される。一部の実施形態例では、コードの脆弱性及びマップされたシステム応答の少なくとも１つがコンピュータアプリケーションのロード時に又は実行時にコードアナライザによって自動で決定される。システム及び方法は、次にコンピュータアプリケーションのコードの脆弱性にアクセスするイベントを検出する。

【００１３】

システム及び方法は、イベントを検出することに応答して、アクセスされるコードの脆弱性にメモリ内のテーブル内でマップされるシステム応答を決定する。一部の実施形態では、脆弱なコードに不適切にアクセスすること等、アプリケーションによる不適切な操作に応答して例外がトリガされる。システム及び方法は、カーネルモード例外ハンドラを上書きするために例外ハンドラをインストルメント（instrument）する。システム及び方法はトリガされる例外を傍受し、インストルメントされた例外ハンドラによってコードの脆弱性を関連付ける。これらの実施形態では、システム及び方法がテーブル内の関連するコードの脆弱性をインストルメントされた例外ハンドラによって照会することを含む。照会することは、コードの脆弱性にマップされるシステムコールバックルーチンとして構成されるシステム応答を返す。アクセスされたコードの脆弱性をイベントが利用するのを防ぐために、システム及び方法は決定されたシステム応答を実行する。例えばシステム及び方法は、コードの脆弱性をイベントが利用するのを防ぐための命令を開始するためにシステムコールバックルーチンを実行する。

【００１４】

システム応答（例えばシステムコールバックルーチンによって開始される命令）は、システムログ内に誤りとしてコードの脆弱性のアクセスのログをとること、アクセスされたコードの脆弱性を含むアプリケーションプロセスのイメージをダンプすること、コードの脆弱性のアクセス前のコンピュータアプリケーションのコピーを復元すること、メモリから１つ又は複数の修復パッチを動的にロードしてコンピュータアプリケーションを再開することなしにコードの脆弱性を含む少なくとも１つのコンピュータルーチンを修正すること、アクセスされたコードの脆弱性に基づいて終了する（例えばクラッシュする）までコンピュータアプリケーションを実行し続けること、及びコンピュータアプリケーションを先回りして終了することのうちの１つ又は複数を含み得る。

【００１５】

図面の簡単な説明

添付図面に示す本発明の実施形態例についての以下のより具体的な説明から上記の内容が明らかになり、図中、同様の参照文字は異なる図面を通して同じパーツを指す。図面は必ずしも縮尺通りではなく、本発明の実施形態を示すことを重視している。

【図面の簡単な説明】

【００１６】

【図１】本開示の一部の実施形態によるアプリケーション基盤の一例を示す。

【図２】一部の実施形態による、図１のアプリケーション基盤との利用者の対話の一例を示す。

【図３Ａ】クライアントによって実行されるロード時操作の流れ図の一例を示す。

【図３Ｂ】本開示の実施形態における、メモリ破損によって促進される悪意ある攻撃を防ぐための方法（及びシステム）の一例の流れ図を示す。

【図３Ｃ】本開示の実施形態における、メモリ破損を引き起こすためにコードの脆弱性を利用する悪意ある攻撃を防ぐための方法（及びシステム）の一例の流れ図を示す。

【図４Ａ】一部の実施形態によるメモリベースの攻撃のブロック図である。

【図４Ｂ】一部の実施形態による、バイナリ仮想パッチングアプリケーション／プローブ（「ｂｖｐＰｒｏｂｅ」としても知られている）に関連するスタック破損検出機能の第１の段階のブロック図である。

【図４Ｃ】一部の実施形態による、バイナリ仮想パッチングアプリケーション／プローブ（「ｂｖｐＰｒｏｂｅ」としても知られている）に関連するヒープ破損検出機能の第１の段階のブロック図である。

【図５Ａ】一部の実施形態による、静的若しくは動的リンクライブラリ（ＤＬＬ）又は共用オブジェクトの個々のパッチのリアルタイムのパッチングを示す。

【図５Ｂ】一部の実施形態による始動時のパッチのローディングを示す。

【図５Ｃ】一部の実施形態による、１つ又は複数の個々のライトパッチをパージする（又は除去する）ことに関連するサイクルを示す。

【図６】一部の実施形態によるパッチングのタイムラインのブロック図である。
【図７Ａ】本開示の実施形態におけるアプリケーション実行時モニタリング及び分析ソリューション（ＡＲＭＡＳ）のブロック図の一例を示す。
【図７Ｂ】図７ＡのＡＲＭＡＳ基盤内でデータを伝送するために使用されるプロトコルデータ単位（ＰＤＵ）の一例を示す。
【図８】本開示の実施形態を実装することができるコンピュータネットワーク又は同様のデジタル処理環境を示す。
【図９】図８のコンピュータシステム内のコンピュータ（例えばクライアントプロセッサ／装置やサーバコンピュータ）の内部構造の一例の図を示す。
【発明を実施するための形態】
【００１７】
詳細な説明
本発明の実施形態例の説明を以下に示す。図１に示すような企業用のデータセンタのアプリケーション基盤では、ウェブサーバが利用者から（又はウェブサービス経由で他のマシンから）入力ウェブ要求（例えばＨＴＴＰ要求）を受信する。但し、図１に示す実施形態はウェブアプリケーション基盤又はデータセンタに限定されず、これだけに限定されないが個人向け、企業向け、クラウドベース、及び工業用の制御アプリケーションを含み得る。ウェブ要求に応答してウェブサーバがウェブサービスを認証し、認証に成功した場合は更なるウェブ要求に応答してリモートユーザが企業システムからの情報に（ウェブサーバ、ポータル、及びアプリケーションサーバを介して）アクセスするためのセッションを確立する。ウェブサービスは、企業システムのコンピュータ上で実行されるアカウントサービス、貿易金融、財務会計、文書管理、制限なしに他の任意のアプリケーション等の様々なアプリケーションからの情報にアクセスすることができる。
【００１８】
ウェブサーバは、セッションに関するユーザ及びセッション（即ち状態）関連データを内部に保持するが、入力要求を処理するために１つ又は複数のアプリケーションサーバに転送するとき、そのデータを次のサーバに伝えることはない。つまりユーザ及びセッションデータは、非武装地帯内の図１のウェブアプリケーション基盤のウェブサーバにおいて終了される。その後、ウェブ要求に対する応答をアプリケーションサーバがウェブサーバに送信すると、ウェブサーバはユーザ及びセッションデータを参照してどの利用者（利用者のユーザ名又はＩＰアドレスによって識別される）に応答を送信するのかを決定する。ウェブサーバは、このやり方で数千のウェブセッションを同時に管理するためにユーザ及びセッションデータを保持することができる。数千のウェブセッション（ユーザ及びセッションデータ）の一部は企業システムのコンピュータ上で実行される様々なアプリケーション内のコードの脆弱性を利用しようと（且つそれらのアプリケーションを破損しようと）試みるハッカーに属する可能性がある。
【００１９】
マルウェア攻撃の概要
ＮＶＤ（National Vulnerability Database）は２０１１年に約４１００件のアプリケーションの脆弱性を、２０１２年には約５３００件のアプリケーションの脆弱性を数え上げた（これらの脆弱性は２３個の攻撃のカテゴリに分けられる）。それらの攻撃のカテゴリの幾つかは不注意又は間違い設定に起因するが、最多数の攻撃のカテゴリは悪意ある行為者が組織の実行中のプロセス内に悪意あるコンテンツを故意に注入し、後でそれを実行させることを含む。そのような悪意あるコンテンツを注入するプロセスは、不十分な入力検証を行う一部の上手く設計されていないコードを識別し、かかるコードを利用することを含む。例えばコードがユーザ入力サイズに関係する検証を欠く場合、そのコードはバッファ誤り（Buffer Error）の攻撃カテゴリに含まれるバッファ誤り型の攻撃を可能にし得る。それらの攻撃では、悪意ある行為者が侵入し、値のコンテンツを突き止め、そのコンテンツを抜きだすために悪意あるコンテンツを注入する。悪意ある行為者は利益を得るためにかかるコンテンツを毀損する場合もある。コンテンツはクレジットカードのデータ、

知的財産、社会保障番号等の機密情報を含む場合がある。悪意ある行為者は最高入札者にその情報を売ることによってその機密情報を使用して利益を得ることができる。

【００２０】

攻撃を検出するシステムの例

図２は、図１のウェブアプリケーション基盤内の対話の一例を示す。（図２に示す）ウェブアプリケーション基盤では、保護されたウェブサーバが（ウェブサービスクライアント経由で）利用者からウェブ要求（例えばＨＴＴＰ要求）を受信する。ウェブ要求内に含まれる情報（例えばＵＲＬ）を使用し、ウェブサーバはウェブサービスの利用者を認証し、認証に成功した場合はウェブサービスの利用者がウェブアプリケーション基盤内のデータにアクセスするための接続（又はセッション）を確立する。

【００２１】

アプリケーションに接続されながら、攻撃者はトランポリンによってメモリ内のコードにアクセスできるようになる可能性がある。しかし、図２のアプリケーション実行時モニタリング及び分析（ＡＲＭＡＳ）機器がアプリケーションと通信してかかるメモリ破損を検出する。かかるメモリ破損を検出すると、ＡＲＭＡＳ機器はアプリケーションをその既知の破損前状態に一時的に復元するためにシャドースタック又は関数ポインタジャンプテーブル（既知の破損前状態を記憶している）を利用する。承認されていないアクセスは悪意ある敵によるセキュリティ攻撃だと宣言することができ、かかる攻撃は攻撃の通知及び誤りを訂正するための修復アクションのためにユーザインタフェース上の管理サーバによって報告され得る。次いで、ハッカーによって引き起こされた破損を修復するために、管理サーバと通信するＡＲＭＡＳ機器が１つ又は複数のパッチを取得し、影響を受けたサーバにその１つ又は複数のパッチをプッシュすることができる。他の実施形態では、破損メモリ（コード命令）に関連する誤りのログをとること、破損メモリを含むアプリケーションプロセスのイメージをダンプすること（例えばコアダンプ）、クラッシュに終わるまで破損アプリケーションが実行し続けることを認めること、制限なしに他の任意のシステム応答等、ウェブサービスにおける他の応答をＡＲＭＡＳ機器が代わりに動的に実行することができる。

【００２２】

図３Ａは、本開示の原理による、マルウェア活動を検出する準備をするために本明細書でモニタリングエージェント（一部の実施形態によれば「解決クライアント」としても知られている）と呼ぶクライアントの一例がロード時に実行することができる操作を示す。一部の実施形態は、米国特許出願公開第２０１４／０５５４６９号（更に２０１４年９月１２日に出願され参照によりその全体を本明細書に援用する国際公開第２０１５／０３８９４４号）の「自動化されたマルウェアの実行時検出（Automated Runtime Detection of Malware）」の中で記載されている１つ又は複数の対応するアプリケーションマップ及び／又はアプリケーションマップデータベース、又は当技術分野で知られている他の技法を含み得る。パス検証エンジンは、マルウェアの実行が開始した時点から数マイクロ秒のうちにマルウェア活動を確実に検出することができるモニタリングエージェントの一部である。モニタリングエージェントはまず整合性を検査し、次いでアプリケーションのモデルを抽出するためにアプリケーションの各モジュールを分析する。アプリケーションのモデルは以下のテーブル、つまりコードテーブル、エクスポートテーブル、Ｖテーブル、その他テーブル、基本ブロックテーブル、ソフトスポットテーブル、メモリオペランドテーブル、遷移テーブル、分解テーブル、及びクリティカルＯＳ関数テーブルを含み得るアプリケーションマップデータベース内に記憶される。図３Ａの実施形態では、アプリケーションマップデータベースがモニタリングエージェントから離れたシステム上に配置される。他の実施形態では、アプリケーションマップデータベースは、アプリケーションが実行されている同一ハードウェア上に、又はモニタリングエージェント及び分析エンジンの両方の外部にあるハードウェア上に保存され得る。モニタリングエージェントは、分析システム上のアプリケーションマップデータベース内に記憶されるデータを送付するために、ストリーミングエンジンを使用してアプリケーションの抽出モデルを解決プロトコルデータ

単位（ＰＤＵ）へとパッケージ化する。

【００２３】

３０２においてモニタリングエージェントがロード時にアプリケーションの個々の実行可能コンポーネントの処理を開始した後、３０４及び３０６においてコンピュータアプリケーションのモジュールごとに同じ操作がループ状に行われる。アプリケーションの各モジュールがメモリ内にロードされると、モニタリングエージェントが所与のモジュールの全ての命令を調べる。アプリケーションファイルのモジュールは、ＰＥ（Portable Executable：移植可能実行ファイル）、ＥＬＦ（Executable and Linkable Format：実行及びリンク可能形式）、ＣＯＦＦ（Common Object File Format：共通目的ファイル形式）等の標準ファイル形式である。この形式では、アプリケーションのモジュールがコードセクション、エクスポート済みデータセクション、ｖテーブルセクション、及び他の多くの追加のセクションを含むセクションとして編成される。アプリケーションの各モジュールがメモリ内にロードされると、モニタリングエージェントがアプリケーションのモデルの一部として関連情報を抽出する。３１４で、モジュールのコードセクションの境界及びアクセス属性をコードテーブル内のアプリケーションマップデータベースに送付し保存する。このテーブル内の各レコードは｛開始アドレス，終了アドレス｝の形式を有する。３３０で、モジュールのコードセクション内の各基本ブロックの境界及び命令数を基本ブロックテーブル内のアプリケーションマップデータベースに送付し保存する。このテーブル内の各レコードは｛開始アドレス，終了アドレス，及び命令数｝の形式である。３１８で、モジュールのエクスポート済みデータセクションの境界及びアクセス属性をエクスポートテーブル内のアプリケーションマップデータベース内に保存する。このテーブル内の各レコードは｛開始アドレス，終了アドレス｝の形式である。３２２で、モジュールのｖテーブルセクション（もしある場合）の境界及びアクセス属性をＶテーブル内のアプリケーションマップデータベースに送付し保存する。このテーブル内の各レコードは｛開始アドレス，終了アドレス｝の形式である。３２６で、モジュールの他の全てのセクションの境界及びアクセス属性をその他テーブル内のアプリケーションマップデータベースに送付し保存する。このテーブル内の各レコードは｛開始アドレス，終了アドレス，及び保護属性｝の形式である。

【００２４】

各モジュールがメモリ内にロードされると、モニタリングエージェントはアプリケーションのモジュールから他のメモリマッピングデータ（３３６）及びソフトスポットデータ（３３４）も抽出する。メモリマッピングデータは、メモリ割当、メモリ割当の解除、及びメモリのクリティカルセグメントへのメモリ書込のための命令を含む。ソフトスポットデータは、ループを実行する命令（例えばＲＥＰ型の演算コードを伴う命令）を含む、大きいメモリバッファ（スポットスポット）を操作するための命令を含む。３３４で、ソフトスポット命令のアドレス及び各メモリ書込のサイズをソフトスポットテーブル内のアプリケーションマップデータベースに送付し保存する。このテーブル内の各レコードは｛アドレス，書込サイズ｝の形式である。このアドレス及び書込サイズは、宛先がメモリオペランドであるメモリ書込命令のために記憶される。３４０で、このデータをメモリオペランド書込テーブル内のアプリケーションマップデータベース内に記憶する。このテーブル内の各レコードは｛発信元アドレス，メモリ書込サイズ｝の形式である。

【００２５】

アプリケーションの各モジュールがメモリ内にロードされると、モニタリングエージェントはモジュールから遷移マッピングデータ（分岐転送又は遷移データ）も抽出する。遷移マッピングデータは、標的アドレスへの遷移命令をその時点において決定できる直接遷移マッピング、又は標的アドレスへの遷移命令が完全に決定されるのを実行時まで阻止する実行時依存性をそれらの命令が有する間接メモリマッピングのためのものであり得る。３２４で、間接遷移が生じる命令の完全な分解を分解テーブル内のアプリケーションマップデータベースに送付し保存する。３２４及び３３２で、抽出された全ての遷移マッピングも遷移テーブル内のアプリケーションマップデータベースに送付し保存する。このテー

ブル内の各レコードは｛発信元アドレス，宛先アドレス｝の形式である。加えて３２０で、オペレータが実行時の前にマップ遷移テーブル内に遷移マッピングデータを手動で追加することができる。マップ遷移テーブル内に手動でレコードを追加するために、オペレータはマルウェアによる遷移テーブルのあり得る改竄を排除するために２要素認証プロセスを使用して自分自身を認証する必要があり得る。

【００２６】

アプリケーションの各モジュールがメモリ内にロードされると、モニタリングエージェントは３０８で整合性についてもアプリケーションをチェックする。一実施形態では、このチェックはモジュールのロード時にコードのＭＤ５ハッシュ等のチェックサムを計算し、そのチェックサムをチェックサムデータベース内に保存されている既知の対応する有効なチェックサムと比較することによって実現される。或いは、信頼できるチェックサム検査サービスを活用することもできる。かかるサービスの利用は、現在ロードしているモジュールのコードがマルウェアによってまだ破損されていないことを保証する。モニタリングエージェントは、整合性チェックが失敗した場合に３１０で警報を発するように構成することができる。

【００２７】

３１２及び３１６において、ロード時に、アクセス許可及びアクセス特権に影響する特定のＯＳ関数及びシステムコールも識別し、それらのアドレスをクリティカルＯＳ関数テーブルに送付し保存する。モニタリングエージェントによって送付される特定のＯＳ関数及びシステムコールは、実行ファイルの実行パスに遠大な影響を有する。これらの管理上の及びクリティカルなＯＳ関数及びシステムコールは、メモリセグメントのアクセス許可を変更し、アクセス特権を増大し、非実行ポリシを変更し、構造化された例外ハンドラ保護を変更し、アドレス空間レイアウト無作為化ポリシを遮断し、メモリの割当及び割当解除を行い、新たなプロセスを作成し、新たなスレッドを作成し、又はデータの暗号化及び復号に関与する。

【００２８】

アプリケーションの各モジュールがメモリ内にロードされると、モニタリングエージェントは、実行時にデータを収集するためにアプリケーションのモジュール内に挿入される命令を追加でインストルメントする。インストルメントされたコードは、動的バイナリ分析エンジン及び／又はバイトコードインストルメンテーションエンジンを使用してアプリケーションのモジュール内に挿入される。３３８で、ループを実行する命令等、マルウェアが攻撃する傾向があるモジュール内の領域にソフトスポット命令をインストルメントして実行時にそれらの領域内の活動を追跡する。３２８で、モジュール内に直接遷移及び間接遷移マッピング命令をインストルメントして実行時に遷移マッピングを含む活動を追跡するためのデータを収集する。３３６で、実行時のメモリ書込活動に関するデータを収集するためにモジュール内にメモリオペランド書込命令をインストルメントする。自己修飾コードがある場合は基本ブロックが実行時に変化し得る。加えて３１２及び３１６で、クリティカルＯＳ関数テーブル内に記憶されるＯＳ関数及びシステムコールが関与する活動に関するデータを収集するためにアプリケーション内に命令をインストルメントする。

【００２９】

ロード時に挿入されるインストルメンテーションの結果、クリティカル情報が実行時に生成され分析のために収集される。遷移マッピングデータに関係するインストルメンテーションがアクセスされると、解決クライアントがスレッドＩＤ、現在の命令アドレス、宛先命令アドレス、及び任意選択的に各汎用レジスタ内に含まれるデータを収集する。命令が実行される前にソフトスポットのインストルメンテーションがアクセスされると、モニタリングエージェントは適切なレジスタによってスレッドＩＤ及びスタックの境界を捕捉する。ソフトスポットのインストルメンテーションが完了すると、モニタリングエージェントがスレッドＩＤ、及びこの書込操作によって更新されるメモリの領域をモニタリングエージェントが推定することを可能にする幾つかの汎用レジスタを捕捉する。コールが実行される前にクリティカルなＡＰＩ又はＯＳコールのインストルメンテーションがアクセ

スされると、モニタリングエージェントがスレッドＩＤ、ＡＰＩ名又はシステムコール番号、及び入力パラメータを捕捉する。コールが実行された後にクリティカルなＡＰＩ又はＯＳコールのインストルメンテーションがアクセスされると、モニタリングエージェントがスレッドＩＤ、ＡＰＩ名又はシステムコール番号、及び戻り値を捕捉する。メモリの割当又は割当解除を行うＯＳ関数又はシステムコール内のインストルメンテーションは、アプリケーションが作成している可能性がある様々なヒープに現在関与しているメモリの領域を追跡するのを助ける。マルウェアがヒープ内の制御構造をオーバランさせたいのかどうかを明らかにするために、このメモリエンベロープを利用して間接メモリ書込実行時の標的を追跡する。加えて、キャッシュを使用して基本ブロックの境界を追跡することにより、分析エンジンは基本ブロックが変化したかどうかを判定することができる。この判定が肯定的である場合、モデルデータベース内の基本ブロックテーブルを更新することができる。

【００３０】

攻撃を防ぐ方法

図３Ｂは、本開示の実施形態における、メモリ破損によって促進される悪意ある攻撃を防ぐための方法（及びシステム）３００の一例の流れ図を示す。一部の実施形態では、システム及び方法がロード時間中にコンピュータアプリケーションのモデルを抽出することができる（３８２）。モデルは、これだけに限定されないが発信元情報（例えば発信元メモリアドレス）、宛先情報（例えば宛先メモリアドレス）、遷移、分岐境界情報、基本ブロック境界情報、コードセグメント境界、インポート及びエクスポートテーブル境界、ジャンプテーブル境界、又は当業者に知られている他の任意の種類のコンピュータルーチン関連情報を含み得る。システム及び方法はコンピュータアプリケーションのモデルを記憶することができる（３８４）。システム及び方法は、実行時にデータを収集するためにコンピュータアプリケーション内に命令を挿入することができる（３８６）。システム及び方法は、コンピュータアプリケーションの記憶済みのモデルに対して実行時に収集したデータを分析して１つ又は複数のセキュリティイベントの検出を行うことができる（３８８）。システム及び方法は１つ又は複数のセキュリティイベントの検出に基づいて、コンピュータアプリケーションに関連する少なくとも１つのアクティブプロセスに関連する少なくとも１つのコンピュータルーチンを修正する（即ちパッチを挿入する）ことができる（３８９）。

【００３１】

方法（及びシステム）３００の実施形態は様々な種類の攻撃を修復する／防ぐ。実行時のプロセスメモリを標的とする攻撃は（一部の実施形態によって修復される）多くの深刻な技術的及び動作的な困難を提起する。例えば殆どのサイバセキュリティソリューションは、攻撃が進行中かどうかを確定的に宣言するのに必要な粒度でプロセスメモリ内の動作を観察する能力を有さない。その結果、ＡＰＴ（持続的標的型攻撃）等の洗練されたメモリベースの攻撃が何年間も検出されない場合がある。通常、脆弱なコードを実行しているプロセスが再開されるまで、脆弱ではないコードを得るためにそのコードをスワップアウトすることはできない。その結果、企業は実行を続けて最も精通していない攻撃者の標的にさえなること、又はリブートし動作及び収益の断絶を被ることの２つの受け入れ難い選択肢を強いられる。洗練されたメモリ破損を利用する攻撃は、アプリケーションの独自のコードがその後実行されるのではなく、敵によって駆動されるコードが実行され始めるようにアプリケーションの制御フローが悪意をもって変更されることから始まり得る。アプリケーションコードが制御を譲るもう１つの結果は、他の攻撃シナリオにおいてアプリケーションがハンドルされない例外をとり、クラッシュし得ることである。この攻撃形式は事実上サービス妨害攻撃である。

【００３２】

図３Ｃは、本開示の実施形態における、メモリ破損を引き起こすためにコードの脆弱性を利用する悪意ある攻撃を防ぐための方法（及びシステム）３９０の一例の流れ図である。方法３９０は、図２のウェブサービス基盤（又は他のシステム基盤）のメモリ内にセキ

ュリティポリシデータベースを作成する（３９１）。セキュリティポリシデータベースは、コンピュータアプリケーションのコードの脆弱性を対応するシステム応答にマップするポリシのテーブルを含む。コードの脆弱性及びマップされたシステム応答はコンピュータアプリケーションに関連する開発チーム（又は他の個人）によって提供されても良く、又はコンピュータアプリケーションのコードをロード時若しくは実行時に分析する（例えばＡＲＭＡＳ機器の）コードアナライザ若しくは他のかかるシステムツールによって自動で又は静的に決定され得る。方法３９０は、コンピュータアプリケーションの脆弱なコード（コードの脆弱性）にアクセスするイベントを検出する（３９２）。例えば脆弱なコードにアクセスすることはメモリアクセス違反又は他のメモリ若しくはシステム違反を引き起こす場合があり、かかる違反は、脆弱なコードを実行しているアプリケーションのスレッド又はプロセスが終了する（即ちクラッシュする）ことを通常引き起こす、ハンドルされないハードウェア又はソフトウェアの例外をトリガする。他の実施形態では、方法３９０は、コンピュータアプリケーションの保存済みのコピーを現在ロードされている又は実行されているコンピュータアプリケーションと比較して、アクセスされたコードの脆弱性に起因するメモリ破損を検出することができる。これらの実施形態では、方法３９０がハードウェア又はソフトウェアの例外を先回りしてトリガし得る。次いで方法３９０は、トリガされたハードウェア又はソフトウェアの例外を傍受する（キャッチする）例外ハンドラをインストルメントすることによってコードの脆弱性のアクセスを検出し、コンピュータアプリケーションを例外が終了させる前にコードの脆弱性を関連付ける。それを行うために、方法３９０はカーネルモード例外ハンドラをインストルメントされた例外ハンドラで上書きし、それにより方法３９０はコンピュータアプリケーションの制御をカーネルから取り戻し、トリガされた例外に対するシステム応答を開始し得る。さもなければ、トリガされた例外に応答してカーネルがコンピュータアプリケーションをクラッシュさせる。

【００３３】

方法３９０は、例外を傍受することに応答してセキュリティポリシデータベース内の例外に関連するアクセスされたコードの脆弱性を自動で照会する１つ又は複数の命令を含むように、インストルメントされた例外ハンドラを提供する。インストルメントされた例外ハンドラによる自動照会により、方法３９０はセキュリティポリシデータベース内に含まれるポリシ内でアクセスされたコードの脆弱性にマップされているシステム応答を決定する（３９３）。決定されたシステム応答がシステムコールバックルーチンの形式でポリシから取得される。インストルメントされた例外ハンドラは、アクセスされたコードの脆弱性が利用されるのを防ぐためにシステム応答（即ちシステムコールバックルーチン）を自動で実行する（３９４）。システムコールバックルーチンは、システムログ内に誤りとしてコードの脆弱性のアクセスのログをとること、アクセスされたコードの脆弱性を含むアプリケーションスレッド又はプロセスのイメージをダンプすること、コードの脆弱性のアクセス前のコンピュータアプリケーションの保存済みのコピーを復元すること、ウェブサービス基盤内のメモリから１つ又は複数の修復パッチを動的にロードしてコードの脆弱性を含むコンピュータルーチンを修正すること、メモリ違反によって終了が生じる（即ちコンピュータアプリケーションがクラッシュする）までコンピュータアプリケーションを実行し続けること、コンピュータアプリケーションを直ちに終了すること、又はコードの脆弱性に関連する他の任意のシステム応答を制限なしに含み得る。

【００３４】

ＡＲＭＡＳプローブ

悪意ある敵がアプリケーションの制御を奪う一例を図４Ａに示す。関数Ｆｏｏ（）のコード（命令のブロック）４０５は、攻撃者によって利用される可能性がある脆弱性を含み得る。図４Ａに示すように、敵は脆弱な関数Ｆｏｏ（）のスタックメモリ４１５を溢れさせることができる。図示のように、Ｆｏｏ（）のスタックメモリ４１５はローカル変数、保存済みベースポインタ、保存済みリターンポインタ４１８、及びパラメータを含む。Ｆｏｏ（）のスタックメモリ４１５の保存済みリターンポインタのメモリ位置が攻撃者によって突き止められオーバランされる場合、Ｆｏｏ（）が終了する（スタックから戻る）と

き、次の命令のアドレスが攻撃者によって保存済みリターンポインタ４１８から取得される場合があり、アプリケーションが攻撃の的になる可能性がある。アプリケーションコードから自らの悪意あるコード（敵のトランポリンコード４１０）内へのトランポリン（ジャンプ）等、攻撃者はかかる技法を使用して別のコードパスへの悪意ある移動を行うことができる。

【００３５】

図４Ａでは、攻撃の標的が命令ポインタレジスタであり得る。攻撃者は他の任意のレジスタも標的とすることができる。多くの関数ポインタがアプリケーションコード内で宣言される場合があり、その値が様々なレジスタから読み出される間接メモリアドレスを使用してロードされ得る。標的とするレジスタ内に悪意あるデータを挿入することにより、攻撃者はプログラムの実行フローを変えることができる。例えば攻撃者は、

```
move rax, rsp
ret
```

等、Ｆｏｏ（）のコード４０５内の命令を標的とするＲＯＰ（リターン指向プログラミング）ガジェットを探すことによってレジスタＥＡＸ（累算器レジスタ）を標的とすることができる。

【００３６】

このことは、敵が制御する値をスタック４１５からＲＡＸ等の他のレジスタ内にロードする効果を有し得る。次に、敵は「call[RAX]」等の命令を実行することにより、Ｆｏｏ（）のアプリケーションコード４０５から自らのコード４１０へのトランポリンをトリガすることができる。どのレジスタが標的とされるのかに関係なく、効果は同じであり得る（敵のコードが実行し始めることができる）。

【００３７】

それに応答して、実施形態はアプリケーションを含む、実行時モニタリング及び分析（ＡＲＭＡＳ）プローブ／アプリケーションを含む。このＡＲＭＡＳプローブはプロセスの全てのスレッド内に並ぶインストルメントされた命令を実行し、プローブが実行時に遭遇する関数コール（例えば上記のmove rax, rsp）やリターン関数コール（例えば上記のret）等の分岐転送命令に関する発信元情報及び宛先情報を収集する。インストルメントされた命令は非常に粒度の細かいスレッドの情報をＡＲＭＡＳプローブに中継し、ＡＲＭＡＳプローブは分岐転送が本当に敵のコード内へのコードトランポリンかどうかを判定することができる。実施形態はバイナリ仮想パッチングアプリケーション／プローブ（又は「ｂｖｐＰｒｏｂｅ」）と呼ばれるよりスマートなＡＲＭＡＳプローブを含み、かかるＡＲＭＡＳプローブは確認のためにＡＲＭＡＳ機器と絶えず接触する制限を克服する。

【００３８】

一部の実施形態によれば、ｂｖｐＰｒｏｂｅの動作は３つの段階（以下に示す段階１から段階３）に分けることができる。

【００３９】

ｂｖｐＰｒｏｂｅの段階１の機能

ｂｖｐＰｒｏｂｅは、スタックベース及び／又はヒープベース攻撃トランポリンを防ぐことができる。図４Ｂは、一部の実施形態によるｂｖｐＰｒｏｂｅに関連するスタック破損検出機能の第１の段階のブロック図である。図４Ｃは、一部の実施形態によるｂｖｐＰｒｏｂｅに関連するヒープ破損検出機能の第１の段階のブロック図である。

【００４０】

図４Ｂに示すように、スタックベースのトランポリンでは、一部の実施形態は最初のステップ（ステップ１）としてコンピュータルーチンの所期の標的アドレスを保存することができる（４２０）。図４Ｂに示すように、次のステップ（ステップ２）で、（例えば図４Ａに関して説明したように）１人又は複数の攻撃者によってメモリ破損４２２が生じ得る。図４Ｂのステップ３に示すように、一部の実施形態はメモリ破損をチェックする（検出する）ことができる（４２４）。ステップ４に示すように、図４Ｂの場合、一部の実施形態は保存済みの標的のコピーを標的アドレスに適用し（４２６）、保存済みの標的を実

行する（４２８）ことによってメモリ破損をバイパスすることができる。一部の実施形態は、これだけに限定されないが米国特許第８，９６６，３１２号「メモリ破損の実行時の検出及び訂正のためのシステム及び方法（System and Methods for Run Time Detection and Correction of Memory Corruption）」等の技法や当技術分野で知られている他の技法を含む、当業者に知られているメモリ破損ハンドリングのための１つ又は複数の技法を使用してメモリ破損に対処することができる。そのため、一部の実施形態はメモリの少なくとも一部のメモリ破損を実行時中に検出し（４２４）、メモリの少なくとも一部をメモリの少なくとも一部のバックアップコピー（即ち保存済みの標的のコピー（４２６））で置換することによってメモリの少なくとも一部のメモリ破損を訂正することができる。このようにして、セキュリティリスクを最小限に抑えながらメモリ破損を適時のやり方で訂正することができる。

【００４１】

同じく図４Ｂに示すように、スタックベースのトランポリンでは、実行時遷移データをＡＲＭＡＳ機器に単純に送付する代わりに、ｂｖｐＰｒｏｂｅはルーチン内への入口点におけるクリティカル状態を保存することができる。図４Ｂのステップ３に示すように、所与のスレッドに対する分岐転送操作が悪意をもって変えられているとｂｖｐＰｒｏｂｅが判定する場合（４２４）、ｂｖｐＰｒｏｂｅはメモリ破損を元通りにすることができ、実行される次の命令のアドレスが取得されるスタック上に保存済みの所期の標的（４２０）を元通り復元することによってコンテキスト上適切な宛先を復元する（４２６～４２８）。

【００４２】

図４Ｃは、ヒープベースのトランポリンに関する破損の検出及びハンドリングを示す。図４Ｃに示すように、図４Ｃのステップ１でアプリケーションが走り実行されるとき、図４Ｃのステップ２でＯｂｊＡのインスタンス（インスタンス１）に敵が制御する入力が書き込まれる。脆弱なコードを含むＯｂｊＡのインスタンス１は、アプリケーションメモリのヒープセクション内で過去に宣言されている。アプリケーションの各オブジェクト（例えばＯｂｊＡやＯｂｊＭ）は通常、読取専用データセクション内の共通仮想ポインタテーブル（ｖｔａｂｌｅ）を指す関数ポインタ１，２．．．Ｎ，．．．，Ｘ等の１つ又は複数の仮想関数ポインタ（ｖｐｔｒ）を有する。一部の実施形態によれば、関数ポインタはアプリケーションメモリの読取専用データセクション内で宣言され得る。敵のデータが（ＯｂｊＡの脆弱なコードに対応する）ｖｐｔｒを上書きすると、図４Ｃのステップ２でｖｐｔｒからｖｔａｂｌｅへのポインタが妨げられる可能性がある。幾らか後の時点において、図４Ｃのステップ３で既知の有効なコードがＯｂｊＡにアクセスすると、既知の有効な関数ポインタを実行する代わりに、図４Ｃのステップ４にあるように敵が選んだ悪意あるコードが実行し始める。ｂｖｐＰｒｏｂｅは、図４Ｃのステップ４で宛先（悪意あるコード）を既知の有効な宛先と比較することによってトランポリンを検出する。

【００４３】

ｂｖｐＰｒｏｂｅの段階２の機能

ｂｖｐＰｒｏｂｅは脆弱性に関する深い見識を収集し、非常に実用的な情報を開発チームに提供することができる。開発チームが脆弱な関数の非脆弱バージョンを開発し試験すると、その関数のための新たなコードをライブラリ内にパッケージ化し、個々のパッチ（「ライト」パッチ）としてリリースすることができる。「ライト」パッチの検査可能且つ符号付きバージョン（signed version）がＡＲＭＡＳダッシュボードにおいて提供されると、ｂｖｐＰｒｏｂｅが実行するようにプロビジョンされ得る１つ又は複数の影響を受けたサーバにそのバージョンをプッシュすることができる。「ライト」パッチは影響を受けたサーバのメモリ内に記憶することができ、それにより（ＡＲＭＡＳダッシュボード上のファイルから繰返しアクセスしたり開発者サイトからネットワーク上で繰返しアクセスしたりするのではなく）「ライト」パッチをメモリからコンピュータアプリケーション内に必要に応じて注入することができる。

【００４４】

個々のパッチ（「ライト」パッチ）を受信すると、ＡＲＭＡＳプローブは対応する個々のパッチ（「ライト」パッチ）に関する構成データを保存することができる。次いで、ユーザコマンドに基づいて（又は自動で）又は誤りに応答してプロセススレッドが中断され得る。図５Ａに示すように、プロセススレッドはバッファ誤り（ＢＥ）脆弱性ヒットに応答して中断され得る。

【００４５】

図５Ａに示すように、プロセススレッドを中断する際、ＡＲＭＡＳプローブはアプリケーションの複雑さに基づき、これだけに限定されないが以下のメカニズムの１つ又は複数を含む手法を使用してアプリケーション内の１つ又は複数のスレッドを停止することができる。メカニズムは、アプリケーションを実行しているハードウェアによって提供される組込（intrinsics）を使用し得る。メカニズム（方法）の最初の例はＳｕｓｐｅｎｄＴｈｒｅａｄ（）ＡＰＩを使用する。この方法ではプロセスが実行中であり（５０５）、分岐実行の脆弱性が攻撃者によってヒットされる（５１０）。この方法は、ＳｕｓｐｅｎｄＴｈｒｅａｄ（）ＡＰＩを使用してプロセス内の全てのスレッドを再帰的に中断する（５１５）。次いでこの方法は、プロセスの中断されたスレッド内に（上記の）ライトパッチＤＬＬを注入し（５２０）、そのことは脆弱な関数にパッチを当て（５２５）、対応する構成ファイルを更新する（５３０）。コードにパッチが当てられると、プローブはＲｅｓｕｍｅＴｈｒｅａｄ（）ＡＰＩを呼び出してプロセスを再開することができる（５３５）。

【００４６】

この手法は、タイムアウトが打ち切られることによる動作上の問題、プロセスの順序が狂って起動することによる競合条件、及び／又は一部の優先順位の低いスレッドが起動時にセマフォを取得することによるデッドロックにマルチスレッドアプリケーションを陥らせる可能性がある。メカニズムの第２の例は、カーネル内に実装される、プロセスを中断することができる１つ又は複数のシステムコール（ＮｔＳｕｓｐｅｎｄＰｒｏｃｅｓｓ（）等）を使用することである。メカニズムの第３の例は、プロセスを一時的に停止するためのデバッガインタフェース（停止のためのＤｅｂｕｇＡｃｔｉｖｅＰｒｏｃｅｓｓ（）及び再開のためのＤｅｂｕｇＡｃｔｉｖｅＰｒｏｃｅｓｓＳｔｏｐ（）を使用することである。

【００４７】

図５Ａに示すように、プロセスが「フリーズ」（即ち中断（５１５））されると、新たなコードを注入し（５２０）（即ち「ライト」パッチＤＬＬを注入し、脆弱な関数に対する「ホット」パッチを行い）、古い脆弱なコードから新たな脆弱ではないコードへのトランポリンを作成することが安全である。これらの悪意のないトランポリンは、これだけに限定されないが以下のメカニズムの１つを使用して実装することができる。メカニズムの第１の例は、新たな「ライト」機能がメモリ内にロードされる場所にジャンプするようにコードセグメント内の機能を直接編集することである。メカニズムの第２の例は、影響を受ける機能がエクスポートされた機能である場合に有用である。メカニズムの第２の例は、古い脆弱なコードが呼び出されるのではなく新たな脆弱ではないコードが呼び出されるようにインポートテーブル内のアドレスを変更する。メカニズムの第３の例は、イベントを作成するコードを挿入し、新たなコードをイベントハンドラに結び付ける。

【００４８】

図５Ａに示すように、ｂｖｐＰｒｏｂｅは中断されたプロセスを起動する（即ちプロセスを再開する（５３５））ことができる。但し同じく図５Ａに示すように、中断されたプロセスを起動する前に、構成ファイル内で構成情報を更新することができる（５３０）。

【００４９】

構成情報は複数のパッチ（即ち複数の「ライト」パッチ）に関する情報を含み得る。パッチに関する構成関連情報を保存するのは重要である。この構成情報はｂｖｐＰｒｏｂｅが１つ又は複数の個々の（「ライト」）パッチを実行可能モジュールの特定のリリースに結び付ける（関連付ける）ことを可能にし得る。次いで、構成が変更されるまで、ｂｖｐＰｒｏｂｅは１つ又は複数の個々の（「ライト」）パッチを始動時にリロードすることが

できる。構成ファイル自体が悪意ある敵の標的になり得る。従って、構成ファイルのコンテンツをエンドポイント固有の「カナリア」によって暗号化することができ、そのファイル整合性チェックサム情報を公開することができ、そのためｂｖｐＰｒｏｂｅがパッチをリロードするために構成ファイルを利用し始める前に構成ファイルを検査することができる。

【００５０】

（例えば図５Ａに示す「ライト」パッチを使用する）仮想パッチングの上記の操作は、プロセスを終了することなしに行えることも重要である。かかる利点はアプリケーションが企業にミッションクリティカルな動作を与えるシナリオにおいて、又はアプリケーションにパッチを当てるコストが大きい場合にとても重要である。

【００５１】

但し、図５Ｂに示すようにフルパッチ（即ち１つ又は複数の個々の又は「ライト」パッチを含み得る集約パッチ）が送られる前にプロセスが再開される場合、「ライト」パッチの挿入が好ましくは始動時に行われる。図５Ｂに示すように、電源再投入が生じるとき、ｂｖｐＰｒｏｂｅアプリケーションがプロセスを開始し（５４０）、（これだけに限定されないがパッチ情報を含む）保存済みの構成情報を取得するために構成ファイルを読み出す（５４２）。次に図５Ｂに示すように、ｂｖｐＰｒｏｂｅアプリケーションはモジュールごとに潜在的に脆弱な関数にパッチを当てるために（５４５）、アプリケーションの各モジュール上に１つ又は複数の個々の（「ライト」）パッチをロードする（５４３）。所望の「ライト」パッチをロードした後、ｂｖｐＰｒｏｂｅはアプリケーションプロセスを実行し始めることができる。

【００５２】

ｂｖｐＰｒｏｂｅの段階３

その後の或る時点において、複数の脆弱性に対する修正で構成されるフルパッチ（即ち１つ又は複数の個々の又は「ライト」パッチを含み得る集約パッチ）を１人又は複数の利用者（ソフトウェア開発チーム等）がリリースし得る。フルパッチは１つ又は複数のライトパッチを含み得る。図５Ｃに示すように、その時点において１つ又は複数のメモリ内の「ライト」パッチを明確に除去することができる。図５Ｃは、一部の実施形態による、１つ又は複数の個々のパッチをパージする（又は除去する）ことに関連するサイクルを示す。

【００５３】

図５Ｃに示すように、１つ又は複数の通信メッセージ（パージパッチメッセージ）をＡＲＭＡＳダッシュボードとＡＲＭＡＳ　ｂｖｐＰｒｏｂｅとの間で送付することができ（５７０）、それによりライトパッチ構成情報を構成ファイル内で更新することができる。ｂｖｐＰｒｏｂｅがパージパッチメッセージを登録し（５７１）、複数のライトパッチに関する情報を含み得る構成ファイルを更新（５７２）して１つ又は複数のモジュールに適用可能なライトパッチの最新のバージョンを反映させる。このようにして、１つ又は複数のモジュールに適用できないライトパッチの１つ又は複数のバージョンをモジュール上にロードしない（即ち「パージする」又はロードしない）ことができる。次いで図５Ｃに示すように、ｂｖｐＰｒｏｂｅがプロセスを開始し（５７３）、各モジュールをロードする（５７４）。モジュールがロードされると、ｂｖｐＰｒｏｂｅは（適用できないライトパッチではなく）適用可能なライトパッチを各モジュール上にロードし（５７５）、そのライトパッチはプロセスの再開前に対応する潜在的に脆弱な関数にパッチを当てる（５７６）。次いでｂｖｐＰｒｏｂｅはプロセスの実行を再開する（５７７）。このようにして一部の実施形態は、ｂｖｐＰｒｏｂｅ、フルパッチがリリースされているライトパッチをパージすることができる。

【００５４】

仮想パッチングのタイムライン

図６は、一部の実施形態によるパッチングのタイムライン、並びに（本明細書に記載の）ｂｖｐＰｒｏｂｅ機能の概要のブロック図である。図６のタイムラインに示すように、

アプリケーションの導入後、図４Ａのメモリベースの攻撃（又は悪意あるトランポリン）によって説明したように関数Ｆｏｏ４０５のスタック４１５等の関数のスタック内でメモリベースの攻撃（即ちゼロデイ攻撃）が生じ得る。図６に示すように、一部の実施形態によれば、（段階１の）ｂｖｐＰｒｏｂｅは、さもなければゼロデイ攻撃を実行している悪意あるトランポリン（敵のトランポリンコード４１０）を阻止することができる。図６に示すように、一部の実施形態によれば、段階２でｂｖｐＰｒｏｂｅは「ライト」パッチ５８０をロードすることができ、段階３でｂｖｐＰｒｏｂｅはリリースされたフルアップグレード５８２（リブートを必要とし得る）の代わりにロードされた「ライト」パッチ５８０を戻す（又は除去し若しくはパージする）ことができる。

【００５５】

例外ハンドリングによる仮想パッチング

他の実施形態では、コードの脆弱性のアクセスをメモリ破損前に検出することができる仮想パッチングアプリケーションをＡＲＭＡＳ機器が実行する。これらの実施形態では、開発チーム（例えばアプリケーションベンダ）がＡＲＭＡＳアプリケーションの使用によって又は動的コード分析若しくは静的コード分析の他の手段（例えばコードアナライザ）によってアプリケーションの脆弱性を突き止めると、アプリケーションのコードの脆弱性がポリシとして構成される。構成される各ポリシは、突き止められたコードの脆弱性とシステムコールバックルーチンとして構成される対応するシステム応答とを含む。突き止めたコードの脆弱性に対するポリシを開発チームが構成するとき、チームは、突き止めたコードの脆弱性に対する推奨されるシステム応答を実行するための対応するシステムコールバックルーチンもプログラムする。推奨されるシステム応答（コールバックルーチンとしてプログラムされる）は、システムログ内に誤りとしてコードの脆弱性のアクセスのログをとること、アクセスされたコードの脆弱性を含むコンピュータアプリケーションスレッド又はプロセスのイメージをダンプすること、コンピュータアプリケーションのスタックの保存済みのコピーを復元すること、ウェブサービス基盤内のメモリから１つ又は複数の修復パッチを動的にロードしてコードの脆弱性を含むコンピュータルーチンを修正すること、メモリ違反によってクラッシュするまでコンピュータアプリケーションを実行し続けること、コンピュータアプリケーションを直ちに終了すること、又はコードの脆弱性に関連する他の任意のシステム応答を制限なしに含む。チームは、推奨される上記のシステム応答の１つを実行するようにプログラムされるコールバックルーチンと共にコンピュータアプリケーションのための既定ポリシを構成することもできる。構成されるポリシは、ＡＲＭＡＳ機器にとってアクセス可能なネットワーク位置に記憶されるセキュリティポリシデータベースのテーブル内に記憶される。

【００５６】

ＡＲＭＡＳ機器は、ネットワークのアプリケーションサーバにおいてハードウェア又はソフトウェア例外ハンドラをインストルメントする。脆弱なアプリケーションコードにイベント（例えばウェブサービス要求）がアクセスすることが原因でハンドルされないメモリアクセス違反又は他のハンドルされないメモリ若しくはシステム違反が生じる場合、アプリケーションを実行しているハードウェア又はソフトウェア（例えばオペレーティングシステム）が例外をトリガする。ＡＲＭＡＳ機器によってインストルメントされたハードウェア又はソフトウェア例外ハンドラはトリガされた例外を傍受し、コードの脆弱性を関連付け、関連するコードの脆弱性を含むポリシを得るためにセキュリティポリシデータベースを照会する。それを行うために、ＡＲＭＡＳ機器はカーネルモード例外ハンドラをインストルメントされた例外ハンドルで上書きして、トリガされた例外に対するシステム応答を開始するためにアプリケーションの制御をカーネルから取り戻す。さもなければ、トリガされた例外に応答してカーネルがアプリケーションをクラッシュさせる。対応するポリシがセキュリティポリシデータベース内に位置する場合、関連するコードの脆弱性のアクセスに応答してインストルメントされた例外ハンドラがポリシのコールバックルーチンを実行する。対応するポリシが見つからない場合、インストルメントされた例外ハンドラは、セキュリティポリシデータベース内のコンピュータアプリケーションのための既定ポ

リシからコールバックルーチンを実行する。
【００５７】
コンピュータアプリケーションの開発チームによって提供されるフルパッチがアプリケーションサーバにダウンロードするために入手可能になるまで、コールバックルーチンはアクセスされたコード内の脆弱性に応答するための仮想パッチとして機能する。以下は例外ハンドルコードの一例である。この例に示す選択肢５は、図５Ａ～図５Ｃに関して上記で説明したようにメモリ内に保存される「ライト」パッチを注入することを含み得る。

```
ｃａｔｃｈ （Ｅｘｃｅｐｔｉｏｎ ｅ）
{
仮想パッチ＝ハンドルされないメモリ違反に関連するコードについてセキュリティポリシデータベースルーチンからプログラム可能なコールバックを取得する；
仮想パッチの実行；
  ／／仮想パッチを実行することは以下の例の１つを含み得る：
    ／／選択肢１：Ｃｏｎｓｏｌｅ．ＷｒｉｔｅＬｉｎｅ（“エラーが発生しました：‘{0}’”，　ｅ）；
    ／／選択肢２：プロセスのイメージをディスクに保存する（コアダンプ）；
    ／／選択肢３：スタックを復元して進む；
    ／／選択肢４：プロセスを続行させクラッシュさせる；
    ／／選択肢５：プロセスを再開することなしにプロセスにパッチを当てる
    ／／.
    ／／.
    ／／.
    ／／選択肢ｘ（制限なし）
}
```

【００５８】
アプリケーション実行時モニタリング及び分析（ＡＲＭＡＳ）基盤
図７Ａは、アプリケーション実行時モニタリング及び分析（ＡＲＭＡＳ）基盤の一例の高レベルブロック図を示す。この基盤はスマートフォン、タブレット、ラップトップ、デスクトップからハイエンドサーバに及ぶ計算装置を含む様々なハードウェア上で構成され得る。図示のように、アプリケーションの性能を改善するために、モニタリングエージェント７０２によって行われるデータ収集を分析エンジン７３７によって行われる分析から分離することができる。この基盤は、マルウェア攻撃に対するこの基盤の保護をハッカーが覆すのを防ぐための高可用性を提供する。モニタリングエージェント７０２はアプリケーションと対話してロード時間及び実行時データを収集する。アプリケーション７０１の基盤は、プロセスメモリ７０３、サードパーティライブラリ７０４、カーネルサービス７０６、及び命令パイプライン７０７を含む。モニタリングエージェント７０２の基盤は、インストルメンテーション及び分析エンジン（インストルメンテーションエンジン）７０５、グラフィカルユーザインタフェース（ＧＵＩ）７１１、クライアントデーモン７０８、構成データベース７０９、ストリーミング及び圧縮エンジン７１０、並びに中央処理装置（ＣＰＵ）７３６を含む。アプリケーション７０１のローカルユーザ又はリモートユーザ７５０は、キーボード、マウス、若しくは同様のＩ／Ｏ装置等の装置によって、又はパイプ、共用メモリ、若しくはソケットによって確立され得る通信チャネルによってネットワーク上でアプリケーションと対話する。それに応答して、アプリケーションプロセス７０３が命令の適切な組を実行のために命令パイプライン７０７内に送付する。アプリケーションは自らのライブラリ、又はlibc.so（Linux）やmsvcrtxx.dll（Windows）等のサードパーティライブラリ７０４を利用することもできる。これらのライブラリからの機能が呼び出されると、これらのライブラリからの適切な命令も実行のために命令パイプライン７０７内に挿入される。加えてアプリケーションは、メモリやカーネル７０６からのファイルＩ／Ｏ等のシステム資源を利用することができる。時間順にまとめられたアプリケーション、ライブラリ、及びカーネルからのこれらの命令のシーケンスは所与の利用者によ

って望まれるアプリケーション機能を届ける。

【００５９】

アプリケーションのコードがメモリ内にロードされ始めると、インストルメンテーションエンジン７０５が幾つかの異なるロード時アクションを実行する。全てのモジュールがロードされると、アプリケーションのインストルメントされた命令が実行時データを生成する。構成データベース７０９から１つ又は複数の構成ファイルを読み出すことにより、クライアントデーモン７０８が７３６におけるＣＰＵ内のインストルメンテーション及び分析エンジン７０５、ストリーミングエンジン７１０、及びＧＵＩ７１１のプロセスを初期設定する。クライアントデーモン７０８は、インストルメンテーションエンジン、ストリーミングエンジン、ＧＵＩ、分析エンジン７３７、及び自らの間の相互通信パイプも初期設定する。クライアントデーモンは、自らを含む任意のモニタリングエージェント７０２のプロセスが応答しなくなる又は終わる場合、そのプロセスが再生成されることも保証する。このように再生成できることは、モニタリングエージェント７０２が高可用性の企業グレード製品であることを保証する。

【００６０】

インストルメンテーション及び分析エンジン７３７は、アプリケーションから収集したロード及び実行時データをストリーミングエンジン内にプッシュする。ストリーミングエンジンはモニタリングエージェント７０２からの生データをＰＤＵへとパッケージ化する。次いでストリーミングエンジンはそのＰＤＵを高帯域幅の低レイテンシ通信チャネル７１２上で分析エンジン７３７にプッシュする。モニタリングエージェント７０２及び分析エンジン７３７が同じマシン上に位置する場合はこのチャネルはメモリバスであり得る。これらのエンティティが異なるハードウェア上だが同じ物理的近傍内に位置する場合はこのチャネルをイーサネット又はファイバによる搬送とすることができ、かかるチャネルはインターネット上でロード及び実行時データを搬送するためにエンティティ間でリモート接続を確立できるようにする。

【００６１】

分析エンジン７３７の基盤は、ネットワークインタフェースカード（ＮＩＣ）７１３、パケットプール７１４、タイムスタンプエンジン７１５、プロセッサファブリック７１６、ハッシングエンジン７１７、ＴＣＡＭエンジン７１８、アプリケーションマップデータベース７１９、及びＲＥＧＥＸエンジン７４０を作り出すスレッドコンテキストデータベース７２０を含む。分析エンジン７３７の基盤は、コンテキスト分析エンジン７２１、イベント及びイベントチェーン７２２、イベント管理エンジン７２３、イベントログ７２４、アプリケーションデーモン７２５、分析エンジン構成データベース７２６、ネットワークインタフェース７２７、ダッシュボード又はＣＭＳ７２８、ＳＭＳ／ＳＭＴＰサーバ７２９、ＯＴＰサーバ７３０、アップグレードクライアント７３１、ソフトウェアアップグレードサーバ７３２、ソフトウェアイメージ７３３、イベント更新クライアント７３４、及びイベントアップグレードサーバ７３５を更に含む。

【００６２】

ＰＤＵがプロトコルヘッダと共にネットワークインタフェースカード７１３において傍受され、そこからＰＤＵがプルされパケットプール７１４内に入れられる。ＰＤＵ内のタイムスタンプフィールドがタイムスタンプエンジン７１５によって埋められる。タイムスタンプフィールドを使用することは、過度に長い時間にわたってパケットプールバッファ内でパケットが立ち往生しないことを保証するのに役立つ。

【００６３】

プロセッサファブリック７１６はパケットバッファからパケットをプルし、アドレスフィールドがハッシュ化され、パケット内の適切な位置において置換される。この操作はハッシングエンジン７１７によって行われる。次いで、プロセッサファブリックがパケットバッファからパケットをその到着順に除去し始める。関連データが抽出されアプリケーションマップデータベース７１９内に記憶されるように、ロード時段階の情報を有するパケットが処理される。実行時段階の情報を有するパケットは図５に従って処理される。プロ

セッサファブリック内のプロセッサの数に基づいて分析エンジン７３７の効率を上げる又は下げることができる。

【００６４】

遷移標的データは、スレッドごとにテーブルを有するスレッドコンテキストデータベース７２０内に保存される。プロセッサファブリックは、遷移及びメモリ領域の探索を行うためにＴＣＡＭエンジン７１８も利用する。プロセッサファブリックはハッシュを使用して検索を行うので、使用される実際の時間は予測可能であり非常に短い。ファブリック内のプロセッサの数を慎重に選ぶことにより、パケットごとのスループットを適切に変えることができる。

【００６５】

分析エンジン７３７が探索を行うとき、分析エンジン７３７は無効な遷移、クリティカル／管理関数若しくはシステムコールの無効な操作を時々見つけ、又は不所望の位置上へのメモリ書込みを見つける場合がある。これらの場合のそれぞれにおいて、分析エンジン７３７がイベント及びイベントチェーンデータベース７２２内に記憶されているポリシによって記述されるプログラム済みの重大度のイベントをイベント管理エンジン７２３に送付する。イベントログデータベース７２４内に生のイベントログが記憶される。ダッシュボードはイベントログにアクセスし、アプリケーションのステータスを表示することもできる。

【００６６】

イベント及びイベントチェーンデータベース７２２内の全てのイベントに修復アクションも関連付けられる。利用者は、一方でイベントを無視することから他方でスレッドを終了することまで、一連のアクションから修復アクションを設定することができる。推奨される修復アクションが、イベント更新クライアント７３４及びイベントアップグレードサーバ７３５を使用してアナリストに推奨され得る。上記の推奨アクションを変えるために、アナリストはダッシュボード７２８をしかるべく使用することができる。ダッシュボードは、モニタされる各アプリケーションの状態を表示し且つアプリケーションを開始することや停止すること等、セキュリティアナリストがアプリケーションを或る程度制御することを可能にするＧＵＩインタフェースを提供する。イベントが生成されると、イベントチェーンが通常の状態からその後の状態に進む。新たな状態に関連する修復アクションをとることができる。修復アクションが非無視アクションを含む場合、ＳＭＳ又はＳＭＴＰサーバ７２９を使用してセキュリティアナリストに通知が送信される。セキュリティアナリストのＳＭＳ／ＳＭＴＰアドレスはＬＤＡＰ又は他のディレクトリプロトコルを使用して求めることができる。ダッシュボードからアプリケーションを開始又は停止するプロセスは高い特権を必要とし、そのためセキュリティアナリストはＯＴＰサーバ７３０を使用して認証しなければならない。

【００６７】

新たなイベントを作成し、重大度及びアナリストに推奨される修復アクションと共にイベント及びイベントチェーンデータベース７２２内にリンクすることもできる。このことは、或るインストレーションにおける新たな攻撃に関する固有のイベント及びイベントチェーンを他のインストレーションに送付することを可能にする。そのために、全ての新たなイベント及びイベントチェーンがイベントアップグレードサーバ７３５内にロードされる。イベント更新クライアント７３４はイベントアップグレードサーバ７３５に周期的に接続し認証して、新たなイベント及びイベントチェーンを取得する。その後、イベント更新クライアントはそれらの新たなイベント及びイベントチェーンをイベント及びイベントチェーンデータベース７２２内にロードする。コンテンツ分析エンジン７２１は、新たなイベントチェーン内にカプセル化された新たな攻撃についてアプリケーションの追跡を開始することができる。

【００６８】

クライアントデーモンの場合と同様に、機器デーモン７２５は分析エンジン７３７上で実行される様々なプロセスを開始する役割を担う。そのために、機器デーモン７２５は分

析エンジン構成データベース７２６から構成情報を読み出す必要がある。デーモンは分析エンジン７３７内の全てのプロセスについてハートビートポールを実行する役割も担う。このことは、分析エンジン３７３エコシステム内の全ての装置が常に最高の動作状態にあることを保証する。ハートビートが３回連続して失われることは、標的のプロセスが応答していないことを示唆する。任意のプロセスが尚早に終了する場合、デーモンは自らを含めそのプロセスを回復させる。

【００６９】

ソフトウェア内の誤りを直すこと等の目的で機器ホスト内の、分析エンジン７３７の、又はクライアントのソフトウェアがアップグレードされることが時々ある。そのために、アップグレードクライアント７３１は最新のソフトウェアを入手することができるソフトウェアアップグレードサーバ７３２を絶えずチェックする。分析エンジン７３７又はクライアント内のエンティティが古いイメージを実行していることをクライアントが見つけ出す場合、クライアントはアナリストが古いイメージをソフトウェアアップグレードサーバ７３２からの新たなイメージでアップグレードすることを可能にする。新たなイメージはシステムイメージ７３３としてまとめられる。そうすることで、試験された互換性のあるイメージを機器又はホストにプロビジョニングできるようになる。分析エンジン７３７又はモニタリングエージェント７０２内のサブシステムのイメージの１つがシステムイメージ内の同じコンポーネントのイメージと一致しない場合、全てのイメージが過去に知られている有効なシステムイメージにロールされる。

【００７０】

ＡＲＭＡＳ通信のためのＰＤＵ

図７Ｂは、図７Ａのモニタリングエージェント７０２と分析エンジン７３７との間でデータを伝送するために使用されるプロトコルデータ単位（ＰＤＵ）の一例を示す。モニタリングエージェント７０２及び分析エンジン７３７が互いに効果的に機能するために、モニタリングエージェント７０２及び分析エンジン７３７はこのＰＤＵを使用して互いに通信する。分析エンジン７３７に伝送するためにアプリケーションの抽出済みモデル及び／又は収集済みの実行時データをパッケージ化するために、このＰＤＵはとりわけモニタリングエージェント７０２によって使用され得る。ＰＤＵは、モニタリングエージェント７０２と分析エンジン７３７との間で伝送される情報の種類ごとにフィールドを含む。ＰＤＵは、アプリケーション提供データセクション、ＨＷ／ＣＶＥ生成セクション、及びコンテンツ分析エンジン又は生データセクションに分割される。

【００７１】

アプリケーション提供データセクションは、様々なレジスタからのデータ並びにこのセクションの様々なフィールド内に配置される発信元アドレス及び標的アドレスを含む。プロトコルバージョンはＰＤＵ７５２のバージョン番号を含む。プロトコルバージョンは時間と共に変化するので、発信元及び宛先は互いに通信を継続できる必要がある。この８ビットフィールドは、発信元エンティティによって生成されるパケットのバージョン番号を記述する。現在未使用の予備フィールド７５６がプロトコルバージョンフィールドの後に続く。

【００７２】

アプリケーション提供データセクションの次のフィールドはメッセージ発信元／宛先識別子７５７、７５３、及び７５４である、図７Ａに示す分析エンジン基盤内でトラフィックを交換するために使用される。図７に示す様々なエンティティはトラフィックを時々交換する。これらの装置の全てがＩＰアドレスを有する又は必要とするわけではなく、従って２つの（ハードウェア及びホスト）照会ルータエンジンは内部でトラフィックをルーティングするためにメッセージ発信元及び宛先フィールドを使用する。一部のメッセージは分析エンジン７３７内のエンティティに辿り着くためにネットワークを横断する必要がある。そのために、エンティティには以下のＩＤが割り当てられる。所与の分析エンジン機器は複数のアクセラレータカードを有し得る。各カードは固有のＩＰアドレスを有し、従って様々なエンティティが固有のＩＤを有する。上述した基盤は複数のアプリケーション

を実行していても良い。各アプリケーションサーバが固有のＩＰアドレスを有するので、対応するモニタリングエージェント側エンティティも固有のＩＤを有する。

【００７３】

モニタリングエージェント側エンティティ
１．ＧＵＩ
２．インストルメンテーション及び分析エンジン
３．クライアントメッセージルータ
４．ストリーミングエンジン
５．クライアント側デーモン
６．ＣＬＩエンジン
７．クライアント監視
８．クライアント圧縮ブロック
９．クライアントiWarpイーサネットドライバ（１００Ｍｂ／１Ｇｂ／１０Ｇｂ）
ＰＣＩカードごとのエンティティ（開始アドレス＝２０＋ｎ＊２０）
２０．セキュラライザＴＯＥブロック
２１．セキュラライザＰＣＩブリッジ
２２．解凍ブロック
２３．メッセージ検査ブロック
２４．パケットハッシングブロック
２５．タイムスタンプブロック
２６．メッセージタイムアウトタイマブロック
２７．統計カウンタブロック
２８．セキュラライザ照会ルータエンジン
２９．セキュラライザアシスト
セキュラライザホストエンティティ
２００．セキュラライザＰＣＩｅドライバ
２０１．ホストルーティングエンジン
２０２．コンテンツ分析エンジン
２０３．ログマネージャ
２０４．デーモン
２０５．ウェブエンジン
２０６．監視
２０７．ＩＰＣメッセージングバス
２０８．構成データベース
２０９．ログデータベース
ＳＩＥＭコネクタ
２２０．ＳＩＥＭコネクタ１－Virsec Dashboard
２２１．ＳＩＥＭコネクタ２－HP ArcSight
２２２．ＳＩＥＭコネクタ３－IBM QRadar
２２３．ＳＩＥＭコネクタ４－Alien Vault USM
セキュラライザ基盤エンティティ
２３０．Virsec dashboard
２３１．ＳＭＴＰサーバ
２３２．ＬＤＡＰサーバ
２３３．ＳＭＳサーバ
２３４．資格サーバ
２３５．データベースバックアップサーバ
２３６．ＯＴＰクライアント
２３７．ＯＴＰサーバ
２３８．チェックサムサーバ

２３９．チケッティングサーバ
２４０．Virsec規則サーバ
２４１．Virsec更新サーバ
オールユーザアプリケーション
２５５．ユーザアプリケーション－照会を発行するアプリケーションを識別するためにアプリケーションＰＩＤが使用される。
【００７４】
アプリケーション提供データセクションの別のフィールドは、伝送されているデータの種類を示すメッセージタイプフィールド７５５である。最上位レベルにおいて、様々なローカルなモニタリングエージェント側エンティティ間、分析エンジン機器側エンティティ間、及びクライアント側エンティティと機器側エンティティとの間を流れる３つの異なる種類のメッセージがある。更に、ネットワークを移動する必要のあるメッセージはＯＳＩモデル及び他のプロトコルに適合する必要がある。
【００７５】
アプリケーション提供データセクションの次のフィールドは、パケットに関するシーケンス識別子を含むパケットシーケンス番号フィールド７７９である。ストリーミングエンジンは損失パケットに対する誤り回復を行う。そのために、ストリーミングエンジンはパケットを一意に識別する必要がある。インクリメントする符号付きの６４ビットパケットシーケンス番号がストリーミングエンジンによって挿入され、残りの分析エンジン基盤を単純に通過する。シーケンス番号が６４ビット境界で完了すると、シーケンス番号は０から再開し得る。ハートビートメッセージやログメッセージ等の非アプリケーションパケットの場合、パケットシーケンス番号は－１とすることができる。
【００７６】
アプリケーション提供データセクションは、暗号化目的で使用されるカナリアを含むカナリアメッセージフィールド７６１も含む。モニタリングエージェント７０２及び分析エンジン７３７は、アプリケーション起動時間、ＰＩＤ、ライセンスストリング、許可されたユーザ名等の幾らかの一般的だが新しい性質の情報からカナリアを計算するやり方を知っている。
【００７７】
加えてアプリケーション提供データセクションは、全てのメッセージ内で使用される汎用フィールドを含む。全般的なアプリケーションデータを保持する、アプリケーション発信元指示アドレス７８０、アプリケーション宛先指示アドレス７５８、メモリ開始アドレスポインタ７５９、メモリ終了アドレスポインタ７６０、アプリケーションＰＩＤ７６２、スレッドＩＤ７６３、分析エンジン到着タイムスタンプ７６４、及び分析エンジン出発タイムスタンプ７６５フィールド。
【００７８】
ＰＤＵはＨＷ／ＣＡＥ生成セクションも含む。分析を容易にし且つ固定の時間予算を保つために、分析エンジン７３７は発信元及び宛先アドレスフィールドをハッシュし、処理の前にＰＤＵを更新する。ＰＤＵのＨＷ／ＣＡＥ生成セクションは、後で使用するためにハッシュデータが配置される場所である。このセクションは、ハッシュアプリケーション発信元指示アドレス７６６、ハッシュアプリケーション宛先指示アドレス７６７、ハッシュメモリ開始アドレス７６８、及びハッシュメモリ終了アドレス７６９のフィールドを含む。加えてＨＷ／ＣＡＷ生成セクションは、ハードコード化コンテンツ開始マジックヘッダ、ＡＰＩ名マジックヘッダ、コールコンテキストマジックヘッダ、及びコール生データマジックヘッダを含む、カナリアに関係する他のフィールド７７１を含む、が全てのＰＤＵパケット内に存在する。
【００７９】
ＨＷ／ＣＡＷ生成セクションは、結果、構成ビット、動作モード、誤りコード、及び動作モードデータを含む他の構成及び誤りデータを識別するためのフィールド７７０も含む。このフィールドの結果部分は、遷移プレイブック、コードレイアウト、メモリ（スタッ

ク又はヒープ）オーバラン、ディープ検査の照会等の分析エンジンの様々な照会に対してブール型の結果を返すようにセグメント化される。このフィールドの構成ビット部分は、圧縮フラグ、デモフラグ、又は併設フラグが設定されたときを示す。このフィールド内にフラグがあることは、パケットを圧縮モードで返すべきかどうかを分析エンジン７３７に知らせる。デモフラグは、システムのための有効なライセンスがないのでシステムがデモモードにあることを示す。このモードでは、ログ及びイベントが全く利用できない。併設フラグは、分析エンジン７３７内でアプリケーションが実行中であることを示し、それによりホスト照会ルータエンジンはアプリケーションに返る必要があるパケットの送信先を明らかにすることができる。このフラグが立てられている場合はパケットがＰＣＩブリッジ経由で送信され、さもなければパケットはＰＣＩカード上のイーサネットインタフェース上で送信される。このフィールドの動作モード部分は、システムがパラノイドモードにあるのか、モニタモードにあるのか、又は学習モードにあるのかを示す。これらのモードについてはこの節の後の部分でより詳細に論じる。最後に、このフィールドの誤りコード部分はシステム内の誤りを示す。誤りコードの最初の８ビットはメッセージ発信元に対応する。残りの１２ビットは各サブシステムによって報告される実際の誤りに対応する。

【００８０】

ＰＤＵは、コンテンツ分析エンジン又は生データも含む。ＯＳライブラリコール及びシステムコールの引数や戻り値等の全ての可変データはＰＤＵのこのセクション内に配置される。このセクション内のデータはアプリケーションから収集されるデータのコンテンツを含み、主にコンテンツ分析エンジン７２１を対象とする。このセクションは、可変サイズのＡＰＩ名又はＡＰＩ番号７７２、コールコンテンツマジックヘッダ７７７、可変サイズコールコンテンツ７７４、コール生データマジックヘッダ７７８、可変サイズ生データコンテンツ７７６、並びに２つの予備フィールド７７３及び７７５を含む。更に、これらのフィールドは管理メッセージについてオーバロードされ得る。

【００８１】

デジタル処理基盤

図８は、本開示の実施形態を実装することができるコンピュータネットワーク又は同様のデジタル処理環境を示す。

【００８２】

クライアントコンピュータ／装置５０及びサーバコンピュータ６０は、アプリケーションプログラム等を実行する処理装置、記憶装置、及び入力／出力装置を提供する。クライアントコンピュータ／装置５０は、通信ネットワーク７０を介して他のクライアント装置／プロセス５０及びサーバコンピュータ６０を含む他の計算装置にリンクすることもできる。通信ネットワーク７０は、互いに通信するためにそれぞれのプロトコル（ＴＣＰ／ＩＰやBluetooth（登録商標）等）を現在使用しているリモートアクセスネットワーク、グローバルネットワーク（例えばインターネット）、世界中のコンピュータの集合、ローカルエリアネットワークや広域ネットワーク、及びゲートウェイの一部であり得る。他の電子装置／コンピュータネットワークアーキテクチャも適している。

【００８３】

クライアントコンピュータ／装置５０は、セキュリティモニタリングエージェントとして構成され得る。サーバコンピュータ６０は、データベースインジェクション攻撃を検出するためにクライアント装置（即ちセキュリティモニタリングエージェント）５０と通信する分析エンジンとして構成され得る。サーバコンピュータ６０は別個のサーバコンピュータでなくても良く、クラウドネットワーク７０の一部であり得る。一部の実施形態では、サーバコンピュータ（例えば分析エンジン）が１組のコンピュータルーチンを分析し、適用される１つ又は複数のパッチを識別し、コンピュータルーチンに１つ又は複数のパッチを適用することができる。クライアント（セキュリティモニタリングエージェント）５０は、サーバ（分析エンジン）６０との間でパッチ及びパッチ要求を通信することができる。一部の実施形態では、クライアント５０が要求及び照会を捕捉し、パッチが必要な破損メモリを検出すると共にパッチを提供するためにクライアント（即ちセキュリティモニ

タリングエージェント）５０上で実行されるクライアントアプリケーション又はコンポーネント（例えばインストルメンテーションエンジン）を含むことができ、クライアント５０はその情報をサーバ（例えば分析エンジン）６０に伝達することができる。

【００８４】

図９は、図８のコンピュータシステム内のコンピュータ（例えばクライアントプロセッサ／装置５０やサーバコンピュータ６０）の内部構造の一例の図である。各コンピュータ５０、６０はシステムバス７９を含み、バスはコンピュータ又は処理システムのコンポーネント間でデータを転送するために使用される１組のハードウェア線である。システムバス７９は本質的にコンピュータシステムの様々な要素（例えばプロセッサ、ディスク記憶域、メモリ、入力／出力ポート、ネットワークポート等）を接続する共用コンジットであり、要素間で情報を転送することを可能にする。システムバス７９に付加されるのが、様々な入出力装置（例えばキーボード、マウス、ディスプレイ、プリンタ、スピーカ等）をコンピュータ５０、６０に接続するためのＩ／Ｏ装置インタフェース８２である。ネットワークインタフェース８６は、ネットワーク（例えば図８のネットワーク７０）に付加される他の様々な装置にコンピュータが接続することを可能にする。メモリ９０は、本開示の実施形態（例えば本明細書に記載のセキュリティモニタリングエージェント、インストルメンテーションエンジン、及び分析エンジンの要素）を実装するために使用されるコンピュータソフトウェア命令９２及びデータ９４用の揮発性記憶域を提供する。ディスク記憶域９５は、本開示の実施形態を実装するために使用されるコンピュータソフトウェア命令９２及びデータ９４用の不揮発性記憶域を提供する。中央処理装置８４もシステムバス７９に付加され、コンピュータ命令を実行できるようにする。

【００８５】

実施形態又は実施形態の側面は、これだけに限定されないがハードウェア回路、ファームウェア、又はソフトウェアを含むハードウェア形式で実装することができる。ソフトウェアによって実装される場合、ソフトウェア又はソフトウェアの命令のサブセットをプロセッサがロードすることを可能にするように構成される任意の非一時的コンピュータ可読媒体上にソフトウェアを記憶することができる。プロセッサはその命令を実行し、本明細書に記載したやり方で動作するように又は器具を動作させるように構成される。

【００８６】

一部の実施形態は、コンピュータルーチンの少なくとも１つをパッチ更新によって非同期的且つ動的に操作することにより、１組のコンピュータルーチンの挙動及び／又はデータを変えることができる。パッチは（これだけに限定されないが）コンピュータルーチンの１つ又は複数に関連する値、入力パラメータ、戻り値、又はコード本体を修正することを含むことができ、それによりコンピュータルーチンの挙動（及び／又はデータ）を変える。

【００８７】

一部の実施形態は、コンピュータアプリケーション及び／又はコンピュータコード内のコンピュータルーチン及び／又は脆弱性の悪意あるハンドリングを検出することにより、コンピュータアプリケーションの質、コンピュータプログラムの機能、及び／又はコンピュータコードの機能上の改善をもたらし得る。一部の実施形態は、予期せぬ挙動及び／又は正しくない挙動を回避するために、１つ又は複数のパッチを導入して不適切に実行されているコンピュータルーチンを訂正し及び／又は置換することができる。そのため、一部の実施形態はコンピュータコードの機能を検出し訂正し、それにより大幅な機能改善をもたらすことができる。

【００８８】

一部の実施形態は、ソフトウェアの機能及びソフトウェアの誤りハンドリング機能のロバスト性を改善することによって技術的問題を解決する（それにより技術的効果をもたらす）。一部の実施形態は、既存の手法を使用するのでは解決が困難であり得るコード破損を検出し修復する技術的問題も解決する（それにより技術的効果をもたらす）。

【００８９】

更に、本明細書ではハードウェア、ファームウェア、ソフトウェア、ルーチン、又は命令をデータプロセッサの特定のアクション及び／又は機能を実行するものとして記載している場合がある。但し、本明細書に含まれるかかる記載は便宜上のものに過ぎず、かかるアクションは計算装置、プロセッサ、コントローラ、又は他の装置がファームウェア、ソフトウェア、ルーチン、命令等を実行することによって実際に生じることを理解すべきである。

【００９０】

流れ図、ブロック図、及びネットワーク図は、更に多くの又は少ない要素を含んでも良いこと、異なるように配置されても良いこと、又は異なるように表現されても良いことを理解すべきである。但し、実施形態の実行を示すブロック図及びネットワーク図並びにブロック図及びネットワーク図の数が特定のやり方で実装されることを特定の実装形態が指図する場合があることを更に理解すべきである。

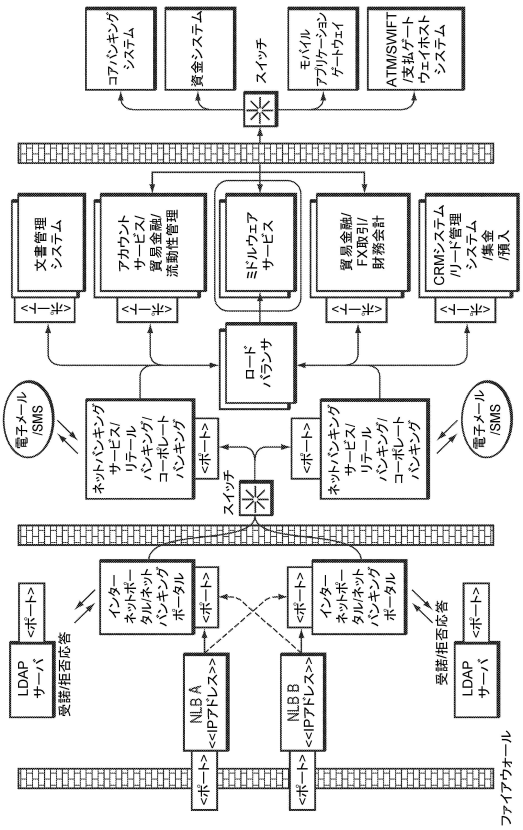
【００９１】

従って、様々なコンピュータアーキテクチャ、物理コンピュータ、仮想コンピュータ、クラウドコンピュータ、及び／又はそれらのものの幾つかの組合せによって更なる実施形態を実装することもでき、そのため本明細書に記載したデータプロセッサは実施形態の限定ではなく例示目的に過ぎない。

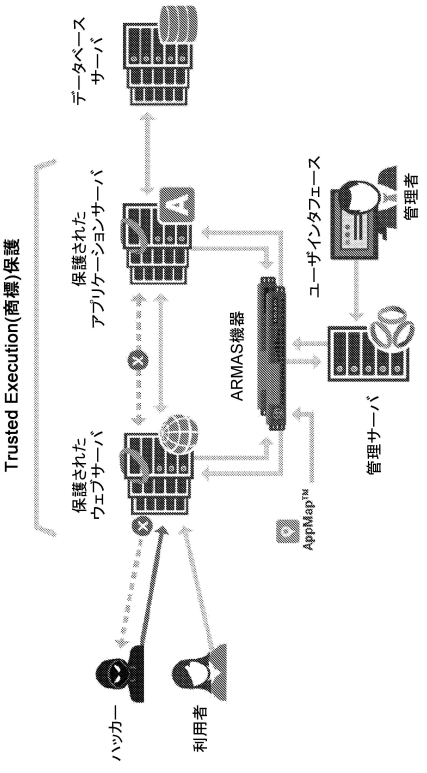
【００９２】

本開示をその実施形態例に関して具体的に示し説明してきたが、添付の特許請求の範囲によって包含される本開示の範囲から逸脱することなしに形式及び詳細の点で様々な変更をその範囲内で加えることができることを当業者なら理解されよう。

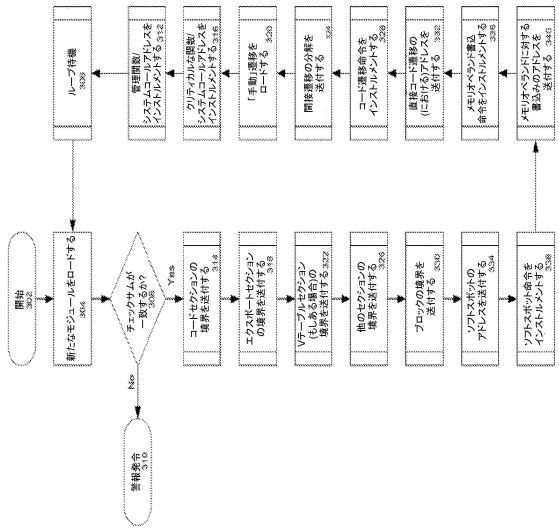
【図１】



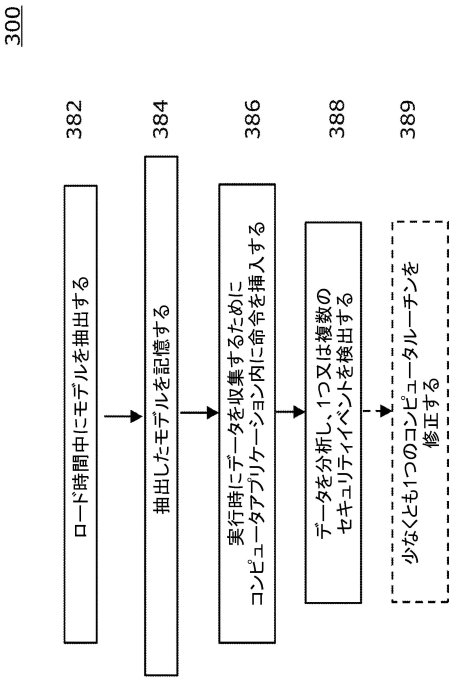
【図２】



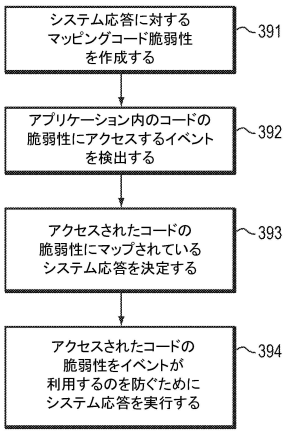
【図３Ａ】



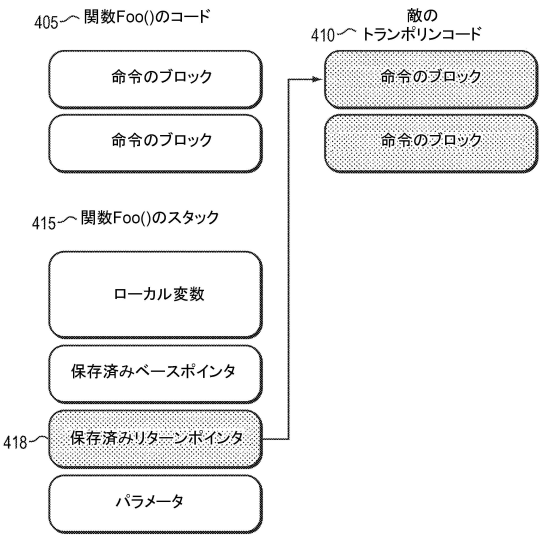
【図３Ｂ】



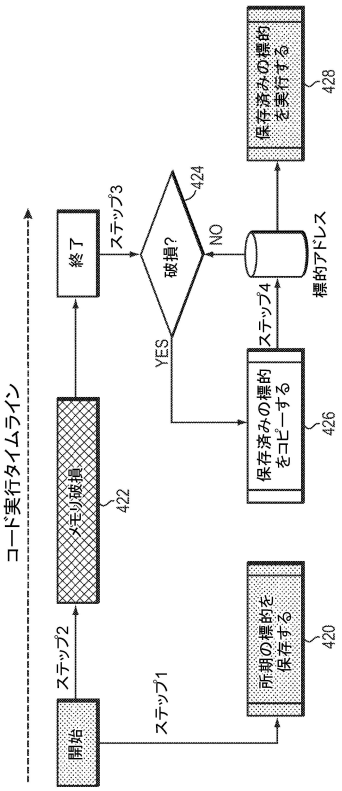
【図３Ｃ】



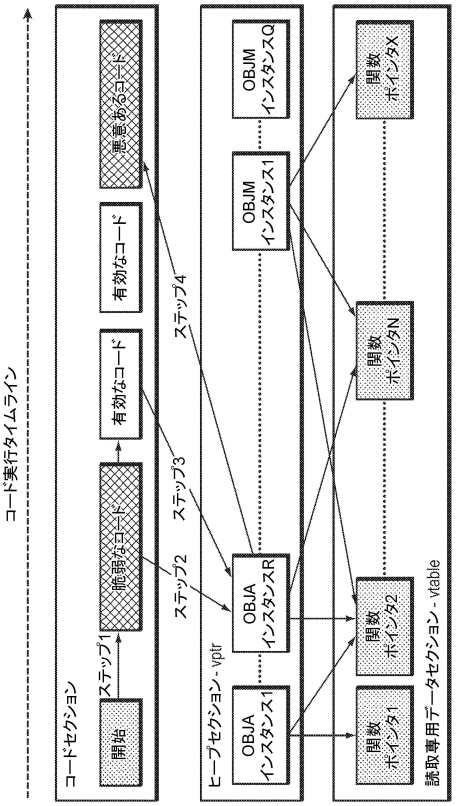
【図４Ａ】



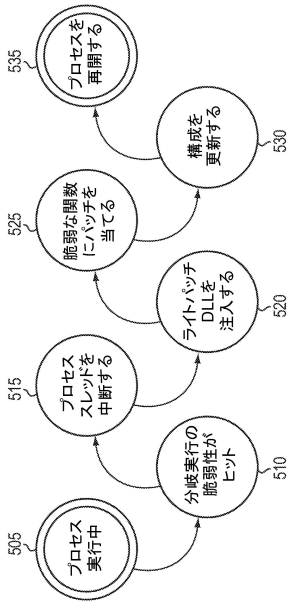
【図４Ｂ】



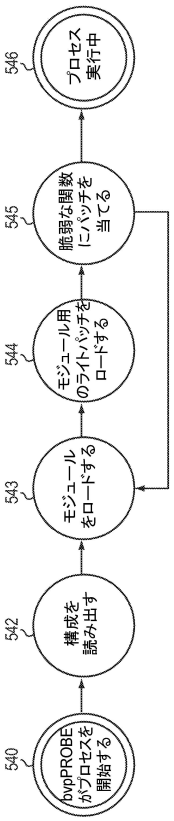
【図４Ｃ】



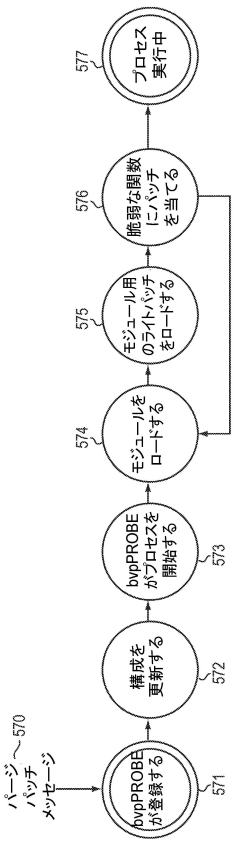
【図５Ａ】



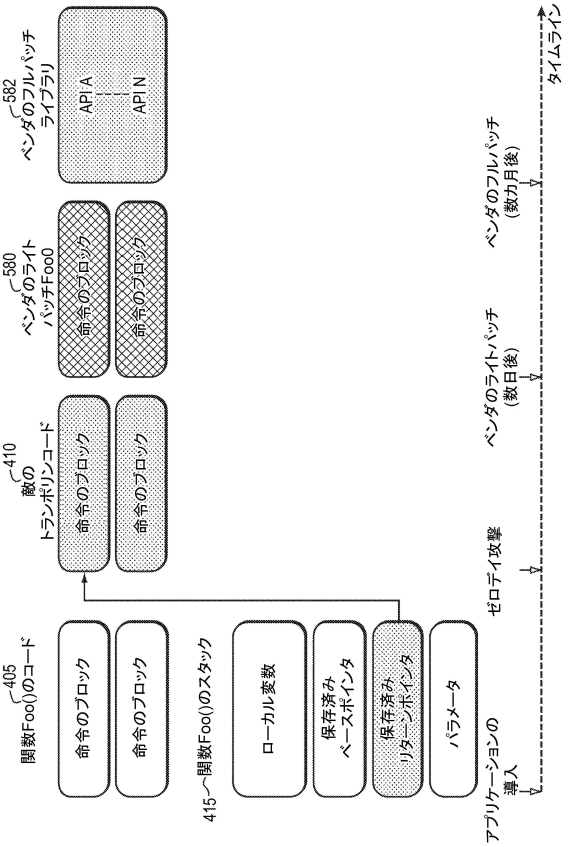
【図５Ｂ】



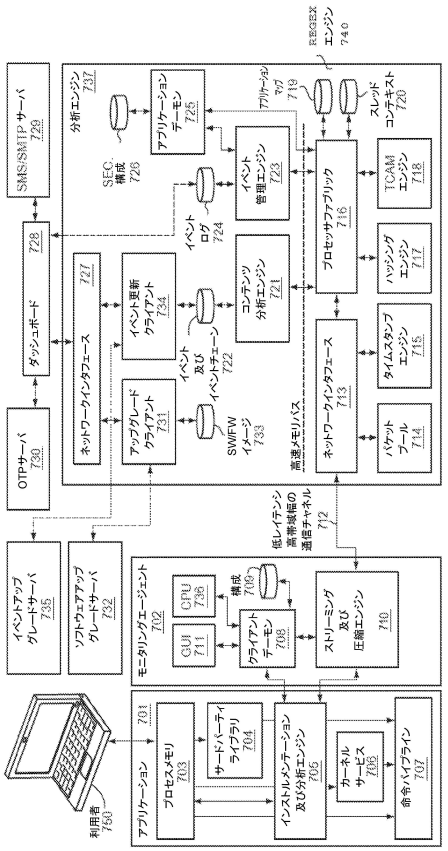
【図５Ｃ】



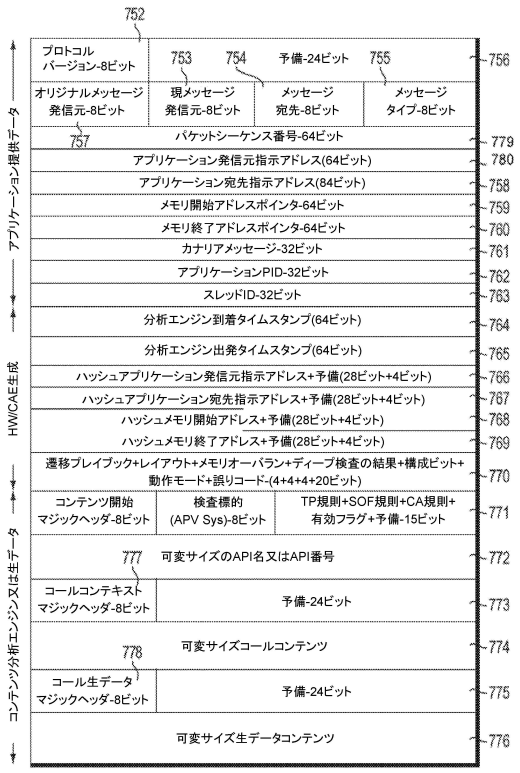
【図６】



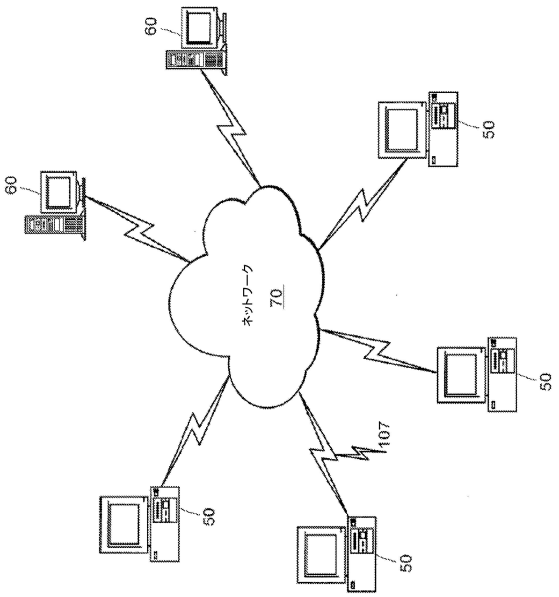
【図７Ａ】



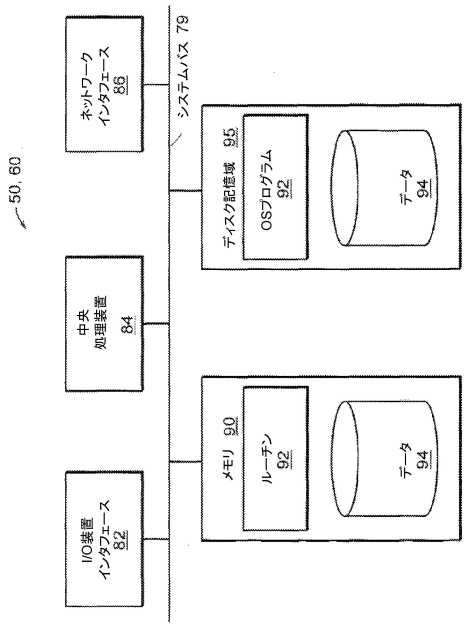
【図７Ｂ】



【図８】


60
60
50
50
50
50
ネットワーク
70
107

【図９】


50, 60
ネットワーク
インタフェース
86
システムバス 79
中央
処理装置
84
I/O装置
インタフェース
82
ディスク記憶域 95
OSプログラム
92
データ
94
メモリ 90
ルーチン
92
データ
94

フロントページの続き

(72)発明者　グプタ，サティヤ，ブラット
アメリカ合衆国，カリフォルニア州　９４５６８，ダブリン，ザック　コート　９６９９

審査官　岸野　徹

(56)参考文献　国際公開第２０１５／０３８９４４（WO，Ａ１）
国際公開第２０１５／２００４６（WO，Ａ１）
特開２００５－２５８４９８（ＪＰ，Ａ）
特表２００９－５０１３６９（ＪＰ，Ａ）
特開２００６－０５３７６０（ＪＰ，Ａ）
特開２０１１－１９８０２２（ＪＰ，Ａ）
国際公開第２０１４／０２１１９０（WO，Ａ１）
特開平０９－２８２１８０（ＪＰ，Ａ）
特開２００７－０５８８６２（ＪＰ，Ａ）
米国特許出願公開第２０１４／０３０４８１５（ＵＳ，Ａ１）
特表２０１６－５３４４７９（ＪＰ，Ａ）
特表２０１７－５２３５１１（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　２１／５２
Ｇ０６Ｆ　２１／５６