(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6777732号
(P6777732)
(45)発行日　令和2年10月28日(2020.10.28)　(24)登録日　令和2年10月12日(2020.10.12)

(51)Int.Cl.　　F I
G06F 21/55 (2013.01)　　G06F 21/55 340
G06F 12/14 (2006.01)　　G06F 12/14 510A

請求項の数 15　(全 24 頁)

(21)出願番号　特願2018-513605(P2018-513605)
(86)(22)出願日　平成28年8月12日(2016.8.12)
(65)公表番号　特表2018-532187(P2018-532187A)
(43)公表日　平成30年11月1日(2018.11.1)
(86)国際出願番号　PCT/US2016/046747
(87)国際公開番号　WO2017/048426
(87)国際公開日　平成29年3月23日(2017.3.23)
審査請求日　令和1年7月18日(2019.7.18)
(31)優先権主張番号　62/219,970
(32)優先日　平成27年9月17日(2015.9.17)
(33)優先権主張国・地域又は機関
米国(US)
(31)優先権主張番号　15/057,336
(32)優先日　平成28年3月1日(2016.3.1)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　507364838
クアルコム，インコーポレイテッド
アメリカ合衆国　カリフォルニア　９２１２１　サン　ディエゴ　モアハウス　ドライブ　５７７５
(74)代理人　100108453
弁理士　村山　靖彦
(74)代理人　100163522
弁理士　黒田　晋平
(72)発明者　スダ・アニル・クマール・ガタラ
アメリカ合衆国・カリフォルニア・９２１２１－１７１４・サン・ディエゴ・モアハウス・ドライヴ・５７７５

最終頁に続く

(54)【発明の名称】コンピューティングデバイスにおけるプロセスに対するソフトウェア攻撃の検出

(57)【特許請求の範囲】
【請求項１】
コンピューティングデバイス上で実行されるプロセスに対するソフトウェア攻撃を検出するための方法であって、
前記プロセスによって利用される複数の仮想メモリ領域の構造的属性を監視するステップであって、前記監視される構造的属性が、
前記複数の仮想メモリ領域のアドレス空間レイアウト、
前記複数の仮想メモリ領域に含まれる仮想メモリ領域の数、または、
前記複数の仮想メモリ領域に含まれる仮想メモリ領域のサイズ
を含む、ステップと、
前記監視される構造的属性を前記複数の仮想メモリ領域の予期される構造的属性と比較するステップと、
前記監視される構造的属性と前記予期される構造的属性との間の前記比較に基づいて、前記監視される構造的属性が前記プロセスの異常な挙動を表すかどうかを決定するステップと
を備える、方法。
【請求項２】
前記監視される構造的属性が異常な挙動を表すという決定に応答して保護動作を開始するステップをさらに備える、請求項1に記載の方法。
【請求項３】

前記プロセスによって利用される前記複数の仮想メモリ領域の前記構造的属性を監視するステップが、
前記アドレス空間レイアウトの変更を監視するステップ、
前記複数の仮想メモリ領域のアクセス許可の変更を監視するステップ、または、
前記複数の仮想メモリ領域の状態遷移履歴を監視するステップ
をさらに備える、請求項1に記載の方法。
【請求項４】
前記プロセスによって利用される前記複数の仮想メモリ領域の前記構造的属性を監視するステップが、
前記プロセスが実行中の現在の仮想メモリ領域に関する情報を記憶するステップと、
前記プロセスによって現在実行されている命令に関する新しい情報を受信するステップと、
前記受信された新しい情報に基づいて、前記現在の仮想メモリ領域からの遷移があったかどうかを決定するステップと、
前記現在の仮想メモリ領域からの前記遷移があったという決定に応答して、前記現在の仮想メモリ領域から新しい仮想メモリ領域への前記遷移を記録するステップと
を備える、請求項1に記載の方法。
【請求項５】
前記プロセスによって現在実行されている前記命令に関する前記新しい情報を受信するステップが、プログラムカウンタ値とプロセス識別子とを受信するステップを備える、請求項4に記載の方法。
【請求項６】
前記プロセスによって利用される前記複数の仮想メモリ領域の構造的属性を監視するステップが、
前記プロセスが実行中の現在の仮想メモリ領域に関する情報を記憶するステップと、
前記複数の仮想メモリ領域上で1つまたは複数の仮想メモリ領域追跡戦略を実施するステップと、
前記1つまたは複数の仮想メモリ領域追跡戦略に基づいて、前記現在の仮想メモリ領域からの遷移があったかどうかを決定するステップと、
前記現在の仮想メモリ領域からの前記遷移があったという決定に応答して、前記現在の仮想メモリ領域から新しい仮想メモリ領域への前記遷移を記録するステップと
を備える、請求項1に記載の方法。
【請求項７】
前記複数の仮想メモリ領域上で前記1つまたは複数の仮想メモリ領域追跡戦略を実施するステップが、前記複数の仮想メモリ領域上で選択的追跡、周期的追跡、日和見的追跡、またはページフォールトに基づく追跡を実施するステップを備える、請求項6に記載の方法。
【請求項８】
前記監視される構造的属性を前記複数の仮想メモリ領域の前記予期される構造的属性と比較するステップが、前記監視される構造的属性にルールのセットを適用するステップを備え、ルールの前記セットが、前記複数の仮想メモリ領域の前記予期される構造的属性に基づく、請求項1に記載の方法。
【請求項９】
前記監視される構造的属性を前記複数の仮想メモリ領域の前記予期される構造的属性と比較するステップが、
モデルを生成するために、前記複数の仮想メモリ領域の前記予期される構造的属性をモデル化するステップと、
前記監視される構造的属性を前記モデルと比較するステップと
を備える、請求項1に記載の方法。
【請求項１０】

コンピューティングデバイスであって、
前記コンピューティングデバイス上で実行されるプロセスによって利用される複数の仮想メモリ領域の構造的属性を監視するための手段であって、前記監視される構造的属性が、
前記複数の仮想メモリ領域のアドレス空間レイアウト、
前記複数の仮想メモリ領域に含まれる仮想メモリ領域の数、または、
前記複数の仮想メモリ領域に含まれる仮想メモリ領域のサイズ
を含む、手段と、
前記監視される構造的属性を前記複数の仮想メモリ領域の予期される構造的属性と比較するための手段と、
前記監視される構造的属性と前記予期される構造的属性との間の前記比較に基づいて、前記監視される構造的属性が前記プロセスの異常な挙動を表すかどうかを決定するための手段と
を備える、コンピューティングデバイス。
【請求項１１】
コンピューティングデバイスであって、
前記複数の仮想メモリ領域を備えるメモリと、
前記メモリに結合され、
前記コンピューティングデバイス上で実行されるプロセスによって利用される前記複数の仮想メモリ領域の前記構造的属性を監視することと、
前記監視される構造的属性を前記複数の仮想メモリ領域の前記予期される構造的属性と比較することと、
前記監視される構造的属性と前記予期される構造的属性との間の前記比較に基づいて、前記監視される構造的属性が前記プロセスの異常な挙動を表すかどうかを決定することと
を行うためのプロセッサ実行可能命令によって構成されたプロセッサと
を備える、請求項10に記載のコンピューティングデバイス。
【請求項１２】
前記プロセッサが、
前記監視される構造的属性が異常な挙動を表すという決定に応答して保護動作を開始するためのプロセッサ実行可能命令によってさらに構成される、請求項11に記載のコンピューティングデバイス。
【請求項１３】
前記プロセスによって利用される前記複数の仮想メモリ領域の前記構造的属性を監視するステップが、
前記アドレス空間レイアウトの変更を監視するステップ、
前記複数の仮想メモリ領域のアクセス許可の変更を監視するステップ、または、
前記複数の仮想メモリ領域の状態遷移履歴を監視するステップ
によって実行されるように、前記プロセッサが、プロセッサ実行可能命令によってさらに構成される、請求項11に記載のコンピューティングデバイス。
【請求項１４】
前記プロセッサが、
前記プロセスが実行中の現在の仮想メモリ領域に関する情報を記憶することと、
前記プロセスによって現在実行されている命令に関する新しい情報を受信することと、
前記受信された新しい情報に基づいて、前記現在の仮想メモリ領域からの遷移があったかどうかを決定することと、
前記現在の仮想メモリ領域からの前記遷移があったという決定に応答して、前記現在の仮想メモリ領域から新しい仮想メモリ領域への前記遷移を記録することと
によって、前記プロセスによって利用される前記複数の仮想メモリ領域の構造的属性を監視するために、プロセッサ実行可能命令によってさらに構成される、請求項11に記載のコ

ンピューティングデバイス。
【請求項１５】
コンピューティングデバイスのプロセッサに、
前記コンピューティングデバイス上で実行されるプロセスによって利用される複数の仮想メモリ領域の構造的属性を監視することであって、前記監視される構造的属性が、
前記複数の仮想メモリ領域のアドレス空間レイアウト、
前記複数の仮想メモリ領域に含まれる仮想メモリ領域の数、または、
前記複数の仮想メモリ領域に含まれる仮想メモリ領域のサイズ
を含む、監視することと、
前記監視される構造的属性を前記複数の仮想メモリ領域の予期される構造的属性と比較することと、
前記監視される構造的属性と前記予期される構造的属性との間の前記比較に基づいて、前記監視される構造的属性が前記プロセスの異常な挙動を表すかどうかを決定することとを備える動作を実行させるように構成されたプロセッサ実行可能ソフトウェア命令を記憶した、コンピュータ可読記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
関連出願
この出願は、2015年9月17日に出願された「Detecting Software Attacks on Processes in Computing Devices」と題する米国仮出願第62/219,970号の優先権の利益を主張し、その内容全体が参照により本明細書に組み込まれる。
【背景技術】
【０００２】
デスクトップコンピュータ、ラップトップ、タブレット、およびスマートフォンなどのモバイル通信デバイスを含む様々なコンピューティングデバイスは、メモリに記憶されたソフトウェア命令に従ってアプリケーションおよびシステムプロセスを実行する。特定のアプリケーションプロセスまたはシステムサービスプロセスは、コンピューティングデバイスに対するより高いアクセス許可(たとえば、ルートアクセス許可)を有し得る。これらのプロセスは、特権プロセスを制御し悪意のあるコードを実行することを目的とした制御ハイジャックソフトウェアの対象となる可能性がある。制御ハイジャックソフトウェアの目的は、シェルを取得しようとしたり、プライベートデータを盗もうとしたりすることであり得る。
【０００３】
制御ハイジャックソフトウェア攻撃には、いくつかの異なる種類がある。たとえば、「スタックスマッシング」は、スタックバッファオーバーフローを作成し、スタックに挿入された悪意のあるコードを実行することを含み得る。ヒープバッファオーバーフローは、ヒープのオーバーフローと同様に動作し、メモリ内の別の場所に挿入された悪意のあるコードの実行を引き起こす。リターン指向プログラミングまたは攻撃は、スタックオーバーフローを発生させ、続いて既存のコードの選択された部分を実行し、ともに接続されると悪意のある目的が達成される。
【発明の概要】
【課題を解決するための手段】
【０００４】
様々な実施形態は、コンピューティングデバイス上で実行されるプロセスに対するソフトウェア攻撃を検出するためのコンピューティングデバイス上で実施される方法を含む。様々な実施形態は、プロセスによって利用される複数の仮想メモリ領域の構造的属性を監視するステップと、監視される構造的属性を複数の仮想メモリ領域の予期される構造的属性と比較するステップと、監視される構造的属性と予期される構造的属性との間の比較に基づいて、監視される構造的属性がプロセスの異常な挙動を表すかどうかを決定するステ

ップとを含み得る。
【０００５】
いくつかの実施形態は、監視される構造的属性が異常な挙動を表すという決定に応答して保護動作を開始するステップをさらに含み得る。いくつかの実施形態では、複数の仮想メモリ領域の監視される構造的属性は、プロセスによって利用される仮想メモリ領域の数、複数の仮想メモリ領域の各々のサイズ、複数の仮想メモリ領域のアドレス空間レイアウト変更、複数の仮想メモリ領域のアクセス許可変更、および/または複数の仮想メモリ領域の状態遷移履歴を含み得る。
【０００６】
いくつかの実施形態では、プロセスによって利用される複数の仮想メモリ領域の構造的属性を監視するステップは、プロセスが実行中の現在の仮想メモリ領域に関する情報を記憶するステップと、プロセスによって現在実行されている命令に関する新しい情報を受信するステップと、新しい情報に基づいて、現在の仮想メモリ領域からの遷移があったかどうかを決定するステップと、現在の仮想メモリ領域からの遷移があったと決定したときに、現在の仮想メモリ領域から新しい仮想メモリ領域への遷移を記録するステップとを含み得る。いくつかの実施形態では、新しい情報は、プログラムカウンタとプロセス識別子とを含み得る。
【０００７】
いくつかの実施形態では、プロセスによって利用される複数の仮想メモリ領域の構造的属性を監視するステップは、プロセスが実行中の現在の仮想メモリ領域に関する情報を記憶するステップと、複数の仮想メモリ領域上で1つまたは複数の仮想メモリ領域追跡戦略を実施するステップと、1つまたは複数の仮想メモリ領域追跡戦略から、現在の仮想メモリ領域からの遷移があったかどうかを決定するステップと、現在の仮想メモリ領域からの遷移があったと決定したときに、現在の仮想メモリ領域から新しい仮想メモリ領域への遷移を記録するステップとを含み得る。いくつかの実施形態では、1つまたは複数の仮想メモリ領域追跡戦略は、選択的追跡、周期的追跡、日和見的追跡、およびページフォールトに基づく追跡を含み得る。
【０００８】
いくつかの実施形態では、監視される構造的属性を複数の仮想メモリ領域の予期される構造的属性と比較するステップは、監視される構造的属性にルールのセットを適用するステップであって、ルールのセットが、複数の仮想メモリ領域の予期される構造的属性に基づく、ステップを含み得る。いくつかの実施形態では、監視される構造的属性を複数の仮想メモリ領域の予期される構造的属性と比較するステップは、複数の仮想メモリ領域の予期される構造的属性をモデル化し、監視される構造的属性をモデルと比較するステップを含み得る。
【０００９】
さらなる実施形態は、上記で要約した方法の動作を実行するためのプロセッサ実行可能命令によって構成されたメモリとプロセッサとを含むコンピューティングデバイスを含む。さらなる実施形態は、コンピューティングデバイスのプロセッサに、上記で要約した方法の動作を実行させるように構成されたプロセッサ実行可能ソフトウェア命令を記憶した非一時的プロセッサ可読記憶媒体を含む。さらなる実施形態は、上記で要約した方法の動作の機能を実行するための手段を含むコンピューティングデバイスを含む。
【００１０】
添付の図面は、本明細書に組み込まれ、本明細書の一部を構成し、例示的な実施形態を示し、本明細書に与えられた一般的な説明および詳細な説明とともに、特許請求の範囲の特徴を説明する役割を果たす。
【図面の簡単な説明】
【００１１】
【図１Ａ】様々な実施形態に従って使用されるコンピューティングデバイスのブロック図である。

【図１Ｂ】様々な実施形態による仮想メモリ領域の構造的属性を分析するための観測/分析ユニットの機能ブロック図である。
【図２】様々な実施形態によるプロセスアドレス空間の図である。
【図３Ａ】スタックバッファオーバーフロー制御ハイジャックソフトウェア攻撃を示す図である。
【図３Ｂ】スタックバッファオーバーフロー制御ハイジャックソフトウェア攻撃を示す図である。
【図４Ａ】ヒープバッファオーバーフロー制御ハイジャックソフトウェア攻撃を示す図である。
【図４Ｂ】ヒープバッファオーバーフロー制御ハイジャックソフトウェア攻撃を示す図である。
【図５】リターンオンプログラミング制御ハイジャックソフトウェア攻撃を示す図である。
【図６】様々な実施形態によるコンピューティングデバイス内の仮想メモリ領域監視システムのブロック図である。
【図７】様々な実施形態による仮想メモリ領域遷移トラッカーの構成要素ブロック図である。
【図８】様々な実施形態による、コンピューティングデバイス上で実行されるプロセスに対するソフトウェア攻撃を検出するための方法を示すプロセスフロー図である。
【図９】様々な実施形態による、コンピューティングデバイス上で実行されるプロセスの仮想メモリ領域遷移を追跡するためのハードウェアベースの方法を示すプロセスフロー図である。
【図１０】様々な実施形態による、コンピューティングデバイス上で実行されるプロセスの仮想メモリ領域遷移を追跡するためのソフトウェアベースの方法を示すプロセスフロー図である。
【図１１】いくつかの実施形態の方法を実施するために適したモバイル通信デバイスの構成要素ブロック図である。
【発明を実施するための形態】
【００１２】
添付の図面を参照して、様々な実施形態を詳細に説明する。可能な場合はいつでも、同一または同様の部分を指すために、図面全体にわたって同じ参照番号が使用される。特定の例および実施形態についてなされる言及は、説明のためのものであり、書かれた記述の範囲または特許請求の範囲を限定するものとはしない。
【００１３】
本明細書で使用される「コンピューティングデバイス」という用語は、セルラー電話、スマートフォン、パーソナルまたはモバイルマルチメディアプレーヤ、携帯情報端末、デスクトップコンピュータ、ラップトップコンピュータ、タブレットコンピュータ、サーバ、スマートブック、スマートウォッチ、パームトップコンピュータ、ワイヤレス電子メール受信機、マルチメディアインターネット対応セルラー電話、ワイヤレスゲームコントローラ、ならびにプログラマブルプロセッサおよびメモリを含む同様の個人または企業の電子デバイスのうちのいずれかまたはすべてを指す。
【００１４】
コンピューティングデバイスは、様々なアプリケーションおよびシステムプロセスを実行し、そのうちのいくつかは、ルートアクセスなどの低レベルのコンピューティングデバイスへの特権アクセスを有し得る。これらのプロセスは、コンピューティングデバイスを低レベルで制御しようとする制御ハイジャックソフトウェア攻撃を受ける可能性がある。これらの攻撃の例には、スタックオーバーフロー攻撃またはヒープバッファオーバーフロー攻撃およびリターンオンプログラミング攻撃が含まれ得る。
【００１５】
各プロセスは、ある内部メモリ構造、またはオペレーティングシステムによって維持さ

れる内部状態の地理を有する。たとえば、各プロセスは、そのプロセスの様々な命令およびデータがメモリに記憶されている場所を示すプロセスアドレス空間を有する。通常、プロセスは同じライブラリのセットをロードし、通常、ライブラリ依存関係は実行時に動的に変化しない。また、プロセスは一般にコードを実行し、そのアドレス空間を予測可能な方法でトラバースする。これは、同様の要求のセットを繰り返し処理するように設計されたシステムサービスに特に当てはまる場合がある。

【００１６】

制御ハイジャックソフトウェア攻撃が発生すると、ターゲットプロセスの内部メモリ構造およびアドレス空間をトラバースする際の挙動が変化し得る。たとえば、悪意のあるソフトウェアは、プロセスの制御フローに変化を生じさせる可能性がある(すなわち、異なる命令のセットが実行されるか、異なる順序で実行される)。コードが実行されている場所からのメモリの領域が異なる場合がある。

【００１７】

概要において、様々な実施形態は、制御ハイジャックソフトウェア攻撃によって引き起こされる異常な実行挙動を検出するために、プロセスのメモリ構造の内部状態または地理を監視するためのシステムおよび方法を提供する。プロセスのプロセスアドレス空間は、データ、機能、または命令を記憶するプロセスアドレス空間内のメモリの連続部分を表すいくつかの仮想メモリ領域(VMR)に分割され得る。コンピューティングデバイス内の観測または分析ユニットは、コンピューティングデバイス上で実行されるプロセスのVMRの様々な構造的属性を監視し得る。監視される構造的属性は、プロセスによって利用されるVMRの数、各VMRのサイズ、VMRのアドレス空間レイアウト変更、VMRのアクセス許可変更、およびVMR状態遷移履歴を含み得る。構造的属性は、各VMR内に記憶されたデータとは独立していてもよい。

【００１８】

次いで、プロセスのVMRの監視される構造的属性が、プロセスのVMRの予期される構造的属性と比較され得る。監視される構造的属性が予期される構造的属性から逸脱している場合、それは悪意のある目的のためにプロセスがハイジャックされていることの表示であり得る。次いで、コンピューティングデバイスは、たとえば、プロセスを終了すること、マルウェア対策プログラムを開始すること、またはオペレーティングシステムの特定の部分をプロセスからロックすることによって、応答として特定の保護動作を行うことができる。

【００１９】

図1Aは、様々な実施形態を実施するために適したコンピューティングデバイス100のブロック図である。コンピューティングデバイス100は、とりわけ、デスクトップコンピュータ、ラップトップ、タブレット、任意のタイプのモバイル電子デバイス、サーバ、あるいは任意のタイプの消費者または企業の電子デバイスであり得る。コンピューティングデバイス100は、ソフトウェア命令を実行するための中央処理装置(CPU)102と、コードおよびデータを記憶するためのメモリ104とを含み得る。メモリ104は、プロセッサ実行可能命令を記憶する非一時的コンピュータ可読記憶媒体であり得る。メモリ104は、オペレーティングシステム106を記憶し得る。いくつかのシステムプロセス108および/またはアプリケーションプロセス110は、オペレーティングシステム106内のコンピューティングデバイス100上で実行されてもよい。プロセス108、110のうちのいくつかは、低レベルのコンピューティングデバイス100に対するより高いアクセス(たとえば、ルートまたは基本入力/出力システム(BIOS)アクセス)許可を有する、特定の特権アプリケーションまたはシステムサービスに属してもよい。

【００２０】

コンピューティングデバイス100はまた、プロセス108、110のVMRの属性を監視するためにオペレーティングシステム106内で実行する観測/分析ユニット112を含み得る。観測/分析ユニット112は、VMRの構造的属性を収集および監視し、監視される構造的属性を、プロセス108、110のVMRの予期される構造的属性と比較し得る。観測/分析ユニット112は、モ

デル化、機械学習、およびルールベースの分析を含む、構造的属性を監視および比較するための1つまたは複数の技法を利用し得る。観測/分析ユニット112は、プロセス108、110のうちの1つまたは複数が異常な挙動を示すときを示す出力を生成し得る。

【００２１】

コンピューティングデバイス100はまた、図1Aに示されていない様々な他の構成要素を含み得る。たとえば、コンピューティングデバイス100は、スピーカ、マイクロフォン、モデム、トランシーバ、加入者識別モジュール(SIM)カード、キーパッド、マウス、ディスプレイスクリーンまたはタッチスクリーン、様々な接続ポート、オーディオまたはグラフィックスプロセッサ、追加のハードドライブ、ならびに当技術分野で知られている多くの他の構成要素などの多数の入力、出力、および処理構成要素を含み得る。

【００２２】

図1Bは、図1Aの観測/分析ユニット112の例示的な論理構成要素および情報フローを示す。観測/分析ユニット112は、CPU102のようなプロセッサ内で実行してもよく、プロセスのVMRの予期および観測された構造的属性を特徴付けるために挙動分析技法を使用するように構成されてもよい。観測/分析ユニット112は、観測ユニット120と、抽出ユニット122と、分析ユニット124と、特徴付けユニット126とを含み得る。

【００２３】

様々な実施形態では、観測/分析ユニット112の全部または一部は、観測ユニット120、抽出ユニット122、分析ユニット124、および特徴付けユニット126の一部として実施され得る。ユニット120～126の各々は、ソフトウェア、ハードウェア、またはそれらの組合せにおいて実施されるスレッド、プロセス、デーモン、サブシステム、または構成要素であり得る。様々な実施形態では、ユニット120～126は、オペレーティングシステムの一部内(たとえば、カーネル内、カーネル空間内、ユーザ空間内など)、別個のプログラムまたはアプリケーション内、専用のハードウェアバッファまたはプロセッサ内、あるいはそれらの任意の組合せ内において実施され得る。いくつかの実施形態では、ユニット120～126のうちの1つまたは複数は、コンピューティングデバイス100の1つまたは複数のプロセッサ上で実行されるソフトウェア命令として実施され得る。

【００２４】

特徴付けユニット126は、プロセスの挙動、およびプロセスによって利用されるVMRの予期される構造的属性を特徴付けるように構成され得る。特徴付けユニット126は、観測されたプロセスの挙動に基づいて少なくとも1つのモデルを生成するために、特徴付けられた挙動および予期される構造的属性を使用し得る。特徴付けユニット126は、観測された挙動を挙動モデルと比較し得る。特徴付けユニット126はまた、観測されたプロセスの挙動およびそれぞれの挙動モデルの他のユニットによって行われた比較を集約し得る。特徴付けユニット126は、集約された比較に基づいて、観測されたプロセスが異常に挙動しているかどうかを決定し得る。特徴付けユニット126は、観測されたプロセスによって利用されるVMRの構造的属性を決定し、観測されたプロセスの挙動を特徴付けるためにそのような情報のいずれかまたはすべてを使用するために、観測ユニット120によって収集された情報を使用し得る。

【００２５】

観測ユニット120は、プロセスの挙動を観測/監視し、観測/監視に基づいてVMRの構造的属性を決定するように構成され得る。構造的属性は、プロセスによって利用されるVMRの数、各VMRのサイズ、VMRのアドレス空間レイアウト変更、VMRのアクセス許可変更、およびVMR状態遷移履歴を含み得る。

【００２６】

観測ユニット120は、観測された構造的属性を含む、収集された観測された挙動データを(たとえば、メモリ書込み動作、関数呼出しなどを介して)抽出ユニット122に通信し得る。抽出ユニット122は、観測された挙動データをログファイルから受信または検索し、観測された構造的属性に基づいて1つまたは複数の挙動ベクトルを生成するためにこの情報を使用するように構成され得る。各挙動ベクトルは、値またはベクトルデータ構造にお

ける観測された構造的属性を簡潔に記述し得る。いくつかの実施形態では、ベクトルデータ構造は一連の数を含み得、その各々は、観測ユニット120によって収集されたリアルタイムデータの部分的または完全な表現を表す。

【００２７】

いくつかの実施形態では、抽出ユニット122は、観測ユニット120によって生成されたログファイルから挙動ベクトルを生成するように構成され得る。挙動ベクトルは、挙動分析システム(たとえば、分析ユニット124)がリアルタイムプロセス挙動およびVMR構造的属性を迅速に認識、識別、または分析することを可能にする識別子として機能し得る。いくつかの実施形態では、抽出ユニット122は、サイズ「n」の挙動ベクトルを生成するように構成され得、その各々は、リアルタイムプロセス挙動およびVMR構造的属性をn次元空間にマッピングする。例示的な実施形態では、抽出ユニット122は、プロセスの挙動を特徴付けるために、プロセスのVMR構造的属性の1つまたは複数の特徴に関するクエリに対する回答を生成するために、特徴付けユニット126内の特徴/決定ノードに入力され得る情報を含むように挙動ベクトルを生成するように構成され得る。

【００２８】

抽出ユニット122は、生成された挙動ベクトルを分析ユニット124に(たとえば、メモリ書込み動作、機能呼出しなどを介して)通信し得る。分析ユニット124は、プロセスによって利用されるVMRの監視される構造的属性が、プロセスが正当である、非正当である、または異常であることを示すかどうかなど、プロセスの観測された挙動を特徴付けるために、挙動ベクトルを分類器モデルに適用するように構成され得る。

【００２９】

分類器モデルは、プロセス活動の特定の特徴または態様を評価するために使用され得るデータおよび/または情報構造(たとえば、特徴ベクトル、挙動ベクトル、構成要素リストなど)を含む挙動モデルであり得る。分類器モデルはまた、プロセスによって利用されるいくつかのVMRを監視するための決定基準を含み得る。分類器モデルは、コンピューティングデバイス100上にあらかじめインストールされてもよく、ネットワークサーバからダウンロードまたは受信されてもよく、観測ユニット120において生成されてもよく、またはそれらの任意の組合せでもよい。分類器モデルは、挙動モデル化技法、機械学習アルゴリズム、または分類器モデルを生成する他の方法を使用することによって生成され得る。

【００３０】

いくつかの実施形態では、分類器モデルは、特定のタイプのプロセス(たとえば、アプリケーションプロセス対システムプロセス)に固有のものであり得る。そのような分類器モデルは、特定のプロセスの挙動を評価することに最も関連すると決定されたプロセス特有の特徴/エントリのみを含む/テストする、集中型データモデルを含み得る。

【００３１】

いくつかの実施形態では、分析ユニット124は、特にプロセス挙動の分析が決定的でない場合、分析ユニット124が評価するプロセスの特徴の粒度または詳細レベルを調整するように構成され得る。たとえば、分析ユニット124は、プロセスの挙動を特徴付けることができないという決定に応答して、観測ユニット120に通知するように構成され得る。これに応答して、観測ユニット120は、監視されているVMR構造的属性を変更するか、および/または、分析ユニット124から送信された通知(たとえば、観測された挙動特徴の分析の結果に基づく通知)に基づいて、その観測の粒度(すなわち、観測された挙動が観測される詳細および/または頻度のレベル)を調整し得る。

【００３２】

観測ユニット120はまた、新しいまたは追加のVMR構造的属性を観測し、さらなる分析/分類のために、新しい/追加の観測された挙動データを抽出ユニット122および分析ユニット124に送信し得る。観測ユニット120と分析ユニット124との間のそのようなフィードバック通信は、観測/分析ユニット112が、観測の粒度を再帰的に増加させる(すなわち、より詳細な、および/またはより頻繁な観測を行う)こと、または、観測されるリアルタイムデータを変更することを可能にし得る。観測/分析ユニット112は、分析ユニット124がプ

ロセスの挙動を信頼性の範囲内または信頼性のしきい値レベルまで評価および特徴付けることができるまで、観測の粒度を増加させるか、または観測されるリアルタイムデータを変更し得る。そのようなフィードバック通信はまた、過度の量の処理、メモリ、またはエネルギーリソースを消費することなしに、観測/分析ユニット112が挙動ベクトルおよび分類器モデルを調整または修正することを可能にし得る。

【００３３】

様々な実施形態では、分類器モデルは、プロセス挙動の特定の特徴に基づいてブーストされた決定株のセットであり得る。ブーストされた決定株は、ちょうど1つのノード(すなわち、1つのテスト質問またはテスト条件)と重み値とを有することができ、データ/挙動の軽い非プロセッサ集約的な2値の分類において使用するのに好適であり得る、1レベル決定木である。挙動ベクトルをブーストされた決定株に適用することで、2値の回答(たとえば、1または0、yesまたはnoなど)がもたらされ得る。たとえば、ブーストされた決定株によってテストされる質問/条件は、デバイスマイクロフォンによって検出された単語もしくは音がRF反応環境の特性であるかどうか、またはデバイスカメラによってキャプチャされた、別のデバイスの画像が、ハザードを生成するRF放出として認識可能であるかどうかを含んでよく、これらに対する回答は2値であってよい。ブーストされた決定株は、そのようなモデルは2値の回答を生成するために重要な処理リソースを必要としないため、効率的である。ブーストされた決定株はまた、高度に並列化可能であり、同時に多数の株が並列に/同時に(たとえば、ユニット、コンピューティングデバイス、またはシステム内の複数のコアまたはプロセッサによって)適用またはテストされることを可能にする。

【００３４】

図2は、コンピューティングデバイス100上で実行中であり得るプロセス(たとえば、プロセス108、110のうちの1つ)のプロセスアドレス空間200の一例を示す。コンピューティングデバイス上のオペレーティングシステムは、通常、コンピューティングデバイス上で実行されるアクティブなプロセスごとに別々のアドレス空間を確立する。プロセスアドレス空間200は、いくつかのアドレス範囲または仮想メモリ領域(VMA)を含み得る。各VMAは、プロセスの所与の部分をマッピングし、メモリにロードするために使用され得る。プロセスによってロードされるコードの量、スタックおよびヒープの使用量、およびロードされるライブラリの数に応じて、プロセスアドレス空間200は疎であってもよく密であってもよい。

【００３５】

プロセスアドレス空間200は、ユーザスタック204用のVMAを含み得る。ユーザスタック204は、プロセスによって作成された一時変数を記憶するために使用されるメモリの領域である。ユーザスタック204は、変数がボトムアップからスタックに記憶され、最初にスタックトップからポップされるファーストイン、ラストアウト(FILO)データ構造として構築され得る。プロセスアドレス空間200はまた、libcライブラリ208および他の共有ライブラリ206用のVMAを含み得る。ライブラリ206、208は、いくつかのプロセスが利用するいくつかの共有機能を含み得る。各プロセスは、ライブラリ206、208のコピーをそのアドレス空間にロードし得る。プロセスアドレス空間200はまた、グローバル変数および他の情報を記憶するために使用され得るヒープ210用のVMAを含み得る。情報は、たとえば、malloc()関数およびfree()関数を使用して、ヒープ210に追加およびヒープ210から削除され得る。プロセスアドレス空間200はまた、メインテキスト212用のVMAを含み得、メインテキスト212用のVMAはプロセスのコード本体を含み得る。

【００３６】

VMAは、複数の仮想メモリ領域(VMR)に細分され得、各VMRは、データ、命令、または機能を記憶するVMA内の連続したアドレス範囲であり得る。たとえば、libcライブラリ208のVMAは、たとえばlibcライブラリ208内の特定の機能を包含し得るVMR208aを含み得る。

【００３７】

コンピューティングデバイスのCPUは、プロセスによって現在実行されている命令のアドレスを含むプログラムカウンタ(PC)214を利用し得る。PC214は、アドレス空間の異なる

部分が呼び出されると、プロセスアドレス空間200の周りをジャンプし得る。たとえば、PC214は、メインテキスト212から開始し、機能が呼び出されるとlibcライブラリ208に移動し得る。次いで、PC214は、スタック204に移動し、機能内で使用される一時変数を読み書きし、次いでメインテキスト212に戻ることができる。

【００３８】

プロセスアドレス空間200は、いくつかの異なるタイプの制御ハイジャックソフトウェア攻撃を受ける可能性があり、これは図3A～図5に示されている。攻撃の1つのタイプは、スタックバッファオーバーフロー攻撃、または「スタックスマッシング」攻撃である。スタックバッファオーバーフロー攻撃を示すためのコードの例を以下に示す。

```
void foo(char *str) {
  char buf[128];
  strcopy(buf, str);
  printf("input string stored on stack\n");
}
```

【００３９】

図3Aは、foo関数が呼び出されたときにロードされ得るスタック302の内容を示す。スタック302は、変数buf[128]に割り振られたメモリ、次いで関数の戻りアドレス、スタックフレームポインタ、および入力strパラメータを含み得る。フロー図304は、foo関数がスタック302を使用するプロセスのメインテキストから呼び出されたときのプロセスの通常の挙動を示す。プロセスはメインテキストから開始し、次いで、strcpy関数を呼び出すためにlibcライブラリ(libc.R1)のVMRに移動し、printf関数を呼び出すためにlibcライブラリ(libc.R3)の別のVMRに移動し、次いでプロセスのメインテキストに戻る。

【００４０】

スタックバッファオーバーフロー攻撃においては、攻撃者の目標は、スタック上のバッファをオーバーフローさせ、特定のコードをスタックに挿入し、挿入されたコードにジャンプするためにスタック上の戻りアドレスを変更することである。図3Bは、悪意のあるソフトウェアによって侵害されたスタック312を示す。悪意のあるソフトウェアは、割り振られた変数buf[128]よりも大きい文字列変数によってfoo関数を呼び出し得、スタック312の他の部分またはVMRを上書きするバッファオーバーフローを引き起こす。入力変数は、挿入されたコード(すなわち、Exec("/bin/sh"))および挿入されたコードを指す戻りアドレスを含み得る。したがって、プログラムカウンタがスタック312の戻りアドレスに到達すると、プログラムカウンタは挿入されたコードにジャンプし、実行する。

【００４１】

フロー図314は、スタック312を使用するプロセスのメインテキストからfoo関数が呼び出されたときのプロセスの異常な挙動を示す。プロセスはメインテキストから開始し、次いでstrcpy関数を呼び出すためにlibcライブラリ(libc.R1)のVMRに移動し得る。しかしながら、strcpy関数の戻りアドレスは、バッファオーバーフローによって上書きされ、今はスタック312内の挿入されたコード(Exec("/bin/sh"))を指している。プログラムカウンタは挿入されたコードにジャンプし、実行する。

【００４２】

別のタイプの制御ハイジャック攻撃は、ヒープバッファオーバーフロー攻撃である。ヒープバッファオーバーフロー攻撃を示すためのコードの例を以下に示す。

```
struct compare {
  char buf[128];
voice (func*) (void);
}
void do_compare (struct compare *comp, char *one, char *two) {
  strcpy(comp->buf, one);
  strcat(comp->buf, two);
  return comp->func(comp->buf, "/data/secretstore/");
```

}

【００４３】

図4Aは、do_compare関数が呼び出されたときにロードされ得るヒープ402の内容を示す。ヒープ402は、変数buf[128]に割り振られたメモリを含み、次いで関数へのポインタを含み得る。フロー図404は、ヒープ402を使用するプロセスのメインテキストからdo_compare関数が呼び出されたときのプロセスの通常の挙動を示す。プロセスはメインテキストから開始し、次いでstrcpy関数を呼び出すためにlibcライブラリ(libc.R1)のVMRに移動し、次いで残りの機能を実行するためにメインテキストの別のVMRに戻ることができる。

【００４４】

ヒープバッファオーバーフロー攻撃においては、攻撃者の目標は、ヒープ上に割り振られたバッファをオーバーフローさせ、メモリの別の部分に特定のコードを挿入し、関数呼出しを挿入されたコードにリダイレクトするために関数ポインタを上書きすることである。図4Bは、悪意のあるソフトウェアによって侵害されたヒープ412を示す。悪意のあるソフトウェアは、割り振られたバッファよりも大きいchar変数によってdo_compare関数を呼び出し得、ヒープ412に記憶された関数ポインタを、挿入されたコードへの別の関数ポインタによって上書きするバッファオーバーフローを引き起こす。したがって、プログラムカウンタがヒープ412内の関数ポインタに到達すると、プログラムカウンタは挿入されたコードにジャンプし、実行する。

【００４５】

フロー図414は、do_compare関数がヒープ412を使用するプロセスのメインテキストから呼び出されたときのプロセスの異常な挙動を示す。プロセスはメインテキストから開始し、次いで、strcpy関数を呼び出すためにlibcライブラリ(libc.R1)のVMRに移動し得る。しかしながら、ヒープ412に記憶された関数ポインタは、メインテキストに戻るのではなく、今は挿入されたコードを指している。プログラムカウンタは挿入されたコードにジャンプし、実行する。

【００４６】

別のタイプの制御ハイジャック攻撃は、悪意のある機能を実行するためにコンピューティングデバイス内の既存のコードの小さな部分を組み合わせる、リターンオンプログラミング攻撃である。既存のコードの小さな部分は、リターンオンプログラミング(ROP)ガジェットと呼ばれ得る。ROPガジェットは、リターン命令で終わる小さなコードシーケンスであり、共有ライブラリ、またはプロセスによってロードされるプロセスアドレス空間のメインテキストに配置され得る。ROPガジェットは、シェルを取得するために、または他の悪意のあるタスクを達成するために、特定の順序で呼び出され得る。

【００４７】

図5は、いくつかのROPガジェットを連続的に呼び出すために侵害されたスタック502を示す。スタック502は、図3Bを参照して説明したのと同様のバッファオーバーフローを介して侵害される可能性がある。この場合、スタック502は、ダミーのスタックフレームポインタとともに、特定の順序でいくつかのROPガジェットへのアドレスを含むように上書きされる。フロー図504は、スタック502から実行するときのプロセスの異常な挙動を示す。このプロセスは、メインテキストから開始し、次いでスタック502に記憶されたアドレスに移動し得る。しかしながら、スタック502内のトップアドレスは、今はROPガジェット1を示しており、ROPガジェット1は、libcライブラリ(たとえば、libc.Ri)内のVMR内に配置され得る。ROPガジェット1を実行した後、プログラムカウンタはスタック502に戻り、次いでlibcライブラリ(たとえばlibc.Rk)内の別のVMRに配置され得るROPガジェット2に移動する。コンピューティングデバイスは、スタック502内に配置された順に各ROPガジェットを実行し得る。

【００４８】

他のタイプの制御ハイジャック攻撃は、スタックオーバーフロー攻撃またはヒープバッファオーバーフロー攻撃と同様に機能し得る、整数オーバーフローに基づく攻撃を含み得る。関数内の整数変数に対するバッファ割振りは、オーバーフローを引き起こす変数より

も小さくなるように割り振られ得る。関数ポインタは、挿入された悪意のあるコードを指すオーバーフローに挿入され得る。

【００４９】

これらのタイプの制御ハイジャックソフトウェア攻撃はすべて、一般に、たとえばプロセスがアドレス空間の様々な部分にスタック、ヒープ、またはポインタを変更することによって、プロセスアドレス空間を割り振り、およびトラバースする方法を変更する。プロセスは、通常、かなり予測可能なルーチンを有し得、したがって、プロセスアドレス空間を割り振るための、またはトラバースするための予測可能な挙動を有し得る。したがって、プロセスのためのアドレス空間が監視され、予期される挙動と比較され得、制御ハイジャック攻撃を示す異常な挙動が検出され得る。

【００５０】

図6は、コンピューティングデバイス内のVMR監視システム600の例示的な実施形態のブロック図を示す。VMR監視システム600は、プロセスによって利用されるいくつかのVMRの監視される構造的属性602～610を入力として受信し、監視される構造的属性をプロセスのVMRの予期される構造的属性と比較する、観測/分析ユニット112を含み得る。比較および分析は、ルールのセットに基づいてもよく、VMRの予期される構造的属性およびプロセスのアドレス空間をモデル化することによって機械学習を通じて達成されてもよい。監視される構造的属性が予期される構造的属性と一致しない場合、観測/分析ユニット112は、出力614をコンピューティングデバイス内の他の構成要素に送信し得る。出力614は、プロセスを終了すること、オペレーティングシステムの特定の部分をプロセスからロックすること、マルウェア対策プログラムを開始すること、または他の動作などの保護動作をとるための命令を含み得る。

【００５１】

プロセスは、いくつかのVMRを含む関連付けられるアドレス空間を有し得る。これらのVMRには、スタック、ヒープ、ライブラリ、およびプロセスのメインテキストの一部が含まれ得る。たとえば、各VMRは、機能、変数、ポインタ、またはプロセスアドレス空間に記憶されたデータまたは命令の他の個別のセグメントに対応し得る。

【００５２】

VMRの構造的属性は、VMRの構造、割振り、地理、または状態遷移履歴を経時的に定義または記述する属性を含み得る。構造的属性はデータに依存しない場合がある。言い換えれば、VMR内に記憶された実際のデータまたはコンテンツは構造的属性として含まれない場合があり、VMRの構造的属性の変更はVMRに記憶されたデータの変更に依存しない場合がある。監視される構造的属性602～610は、観測/分析ユニット112によって監視され得る様々な構造的属性の非限定的な例である。

【００５３】

1つの監視される構造的属性602は、アドレス空間内に存在するVMRの数であり得る。プロセスは通常、通常の実行中に一定数のVMRを利用し得る。制御ハイジャック攻撃は、外部の、あるいは挿入されたコードまたはデータを記憶するために、メモリの新しいセクションを割り振ることを含み得る。したがって、VMRの数の変化は、制御ハイジャックソフトウェア攻撃によって引き起こされる異常な挙動を示し得る。VMRの数は、新しいVMRが作成されるたびに関数(たとえば、do_mmap())を呼び出すカーネルマッパを通じて監視され得る。

【００５４】

別の監視される構造的属性604は、各VMRのサイズであり得る。たとえば、プロセスによって使用される特定の変数に割り振られたメモリは、複数の実行にわたって同じになる可能性があるので、VMRのサイズの変化は、制御ハイジャックソフトウェア攻撃(たとえば、人為的にバッファオーバーフローを引き起こす攻撃)によって引き起こされる異常な挙動を示し得る。VMRのサイズは、カーネルマッパ、malloc()関数への呼出し、およびプロセスアドレス空間のライブラリに新しい関数を追加することを通じて監視され得る。

【００５５】

別の監視される構造的属性606は、プロセスのアドレス空間レイアウト変更であり得る。プロセスは通常、VMRが互いに同じ位置に配置され、レイアウトが複数の実行にわたって変化しないように、特定の内部構造によってそのアドレス空間を配置し得る。したがって、アドレス空間のレイアウトの変更は、制御ハイジャックソフトウェア攻撃によって引き起こされる異常な挙動を示し得る。プロセスのアドレス空間レイアウト変更は、カーネルマッパを通じて監視され得る。

【００５６】

別の監視される構造的属性608は、VMRに対するアクセス許可変更であり得る。各VMRには、一般に複数の実行にわたって変化しないプロセスによってアクセス許可が割り当てられ得る。したがって、VMRのアクセス許可の変更は、制御ハイジャックソフトウェア攻撃によって引き起こされる異常な挙動を示し得る。VMRのアクセス許可の変更は、カーネル仮想メモリマネージャを通じて監視され得、VMRの許可が変更されたときはいつでも呼び出され得る。

【００５７】

別の監視される構造的属性610は、経時的なプロセスのVMR状態遷移履歴であり得る。プロセスは、複数の実行にわたって変化しない予測可能な順序で、様々なVMRを通じてナビゲートし得る。したがって、プロセスが予期されないVMR間の一連の状態遷移に関与している場合、それは制御ハイジャックソフトウェア攻撃によって引き起こされる異常な挙動の表示であり得る。VMR状態遷移履歴は、ハードウェアまたはソフトウェアに基づく解決策を通じて監視され得、これらの解決策は、図7、図9、および図10を参照してさらに詳細に説明される。

【００５８】

図7は、コンピューティングデバイス上で実行されるプロセスのVMR遷移を監視するためのハードウェア実施形態を示す。システム700は、プロセスを実行し、プロセスのためのプログラムカウンタを維持するCPUコア702を含む。システム700はまた、コンピューティングデバイス上で実行される様々なプロセスのVMRのメモリ範囲を維持するVMRカーネルドライバ704を含み得る。システム700はまた、VMRトラッカーハードウェアブロック706を含み得る。VMRトラッカーハードウェアブロック706は、CPUコア702からのプログラムカウンタと、VMRカーネルドライバ704からの現在のプロセスID(PID)およびVMR範囲とを入力として受信するVMR遷移ロガー712を含み得る。VMRトラッカーハードウェアブロック706は、PIDによってインデックス付けされた、コンピューティングデバイス上で実行されるプロセスごとのテーブルを含み得る、いくつかのVMRテーブル710を記憶し得る。各テーブルは、そのプロセスによって割り振られたVMRと、プロセスが実行されている最後の知られているVMRとを記憶し得る。

【００５９】

VMR遷移ロガー712は、周期的に、VMR範囲、PID、および/またはプログラムカウンタを入力として受信し得る。これらの入力は、監視されるべきプロセスと、プログラムカウンタによって指示される現在の命令とを識別する。VMR遷移ロガー712は、監視されたプロセスが同じVMR内に依然として存在するか、または異なるVMRに遷移したか(たとえば、プロセスのメインテキストからライブラリ関数への関数呼出し)を決定するために、入力された情報をVMRテーブル710と比較し得る。VMR遷移があった場合、VMR遷移ロガー712はVMR遷移を記録し、それをデータバッファ708に記憶し得る。VMR遷移およびPIDも、VMRカーネルドライバ704に送信され得る。このようにして、VMRトラッカーハードウェアブロック706は、コンピューティングデバイス上で実行されるプロセスにおけるVMR遷移を検出および監視し得る。

【００６０】

VMRトラッカーハードウェアブロック706は、VMR状態遷移履歴を監視するためのハードウェア実施形態であるが、VMR状態遷移履歴は、ソフトウェアを通じて代替的に監視され得る。VMR状態遷移の一定のソフトウェア監視はリソース集約的であり、コンピューティングデバイス上の他のプロセスの動作に影響を与える可能性がある。しかしながら、VMR

状態遷移を非連続的に監視する方法はいくつかあり、個々にまたは組み合わせて実施され得る。

【００６１】

たとえば、選択的追跡VMR状態遷移においては、関心のある特定の実行プロセスに対してのみ監視が開始され得る。他の実行プロセスは監視されない場合がある。追跡ソフトウェアは、関心のあるプロセスのプログラムカウンタを監視し得る。周期的追跡において、追跡ソフトウェアは、タイマスレッドに従ってプロセスのプログラムカウンタを周期的にサンプリングし得る。マルチCPUコアシステムにおいては、プロセスが実行されていないCPUにタイマスレッドが結合され得る。タイマスレッドの周期は、周期性が特定のプロセス挙動に適応され、VMR状態遷移の欠落の可能性が低減され得るように、調整可能であり得る。

【００６２】

日和見的追跡では、VMRカーネルドライバは、プロセスが実行されているCPUの制御を受信し得る。そのような場合、VMRカーネルドライバはプログラムカウンタをサンプリングし、VMR状態遷移を追跡し得る。たとえば、VMRカーネルドライバは、割込み到着、コンテキストスイッチ、信号生成、およびシステムコール中にプロセッサ制御を受信し得る。ページフォールトに基づく追跡においては、追跡ソフトウェアは非現在のVMRを非実行可能(たとえば、読出し専用)に設定し得、プロセスが新しいVMRにジャンプするたびにページフォールトが生じる。追跡ソフトウェアは、ページフォールトを検出するたびに状態遷移を記録し、新しいVMRを実行可能に設定し、古いVMRを非実行可能として設定し得る。

【００６３】

図8は、様々な実施形態による、コンピューティングデバイス上で実行されるプロセスに対するソフトウェア攻撃を検出するための方法800を示す。方法800は、コンピューティングデバイスのプロセッサ(たとえば、図1Aのコンピューティングデバイス100のCPU102)によって実施され得る。

【００６４】

ブロック802において、プロセッサは、コンピューティングデバイス上で実行されるプロセスの複数の仮想メモリ領域の構造的属性を監視し得る。プロセスは、いくつかのVMRに分割され得る関連付けられるアドレス空間を有し得る。VMRは、スタック、ヒープ、ライブラリ、メインテキスト、またはプロセスのためのアドレス空間の他の部分に見出され得るデータ、命令、または機能を記憶するメモリの連続部分であり得る。監視される構造的属性は、プロセスによって利用されるVMRの数、各VMRのサイズ、VMRのアドレス空間レイアウト変更、VMRのアクセス許可変更、およびVMR状態遷移履歴を含み得る。

【００６５】

ブロック804において、プロセッサは、監視された複数のVMRの構造的属性を、プロセスの複数のVMRの予期される構造的属性と比較し得る。コンピューティングデバイスは、VMR間で割り振り、利用し、遷移する際に、プロセスの予期される挙動(すなわち、制御ハイジャック攻撃からの干渉なしに実行する)を事前に分析している可能性があり、また、プロセスのVMRの予期される構造的属性を定義するモデルまたはルールのセットを構築している。プロセッサは、監視される構造的属性を予期される構造的属性と比較するために、数学的モデル、ルールベースの比較、または他の方法を利用し得る。

【００６６】

決定ブロック806において、プロセッサは、プロセスの複数のVMRの監視される構造的属性が、監視される構造的属性と予期される構造的属性との間の比較に基づいて異常な挙動を表すかどうかを決定し得る。たとえば、プロセッサは、監視される構造的属性と予期される構造的属性との間の類似性を表す値を生成するために数学的モデルを使用し、その値を許容可能な類似度を表すしきい値と比較し得る。値がしきい値を下回っている場合、監視される構造的属性は異常な挙動を表す可能性がある。別の例では、プロセッサは、監視される構造的属性にルールのセットを適用し、監視される構造的属性がすべてのルールを満たすかどうかを決定し得る。監視される構造的属性が1つまたは複数のルールを満たさ

ない場合、監視される構造的属性は異常な挙動を表し得る。

【００６７】

監視される構造的属性が異常な挙動を表していないという決定に応答して(決定ブロック806=「No」)、プロセッサは、プロセスが実行されている限り、複数のVMRの構造的属性を監視し続けることができる(すなわち、ブロック802の動作に戻る)。監視される構造的属性が異常な挙動を表すという決定に応答して(決定ブロック806=「Yes」)、プロセッサはブロック808において保護動作を開始し得る。たとえば、プロセッサは、プロセスを終了する、オペレーティングシステムの特定の部分をプロセスからロックする、マルウェア対策プログラムを開始する、または他の動作を行うことができる。このように、方法800は、プロセスのアドレス空間内のVMRの構造的属性を監視することによって、プロセスに対する制御ハイジャックソフトウェア攻撃を監視する方法を提供する。

【００６８】

図9は、様々な実施形態による、コンピューティングデバイス上で実行されるプロセスのVMR状態遷移履歴のハードウェアベースの追跡のための方法900を示す。方法900は、コンピューティングデバイスのプロセッサ(たとえば、図1Aのコンピューティングデバイス100のCPU102)によって実施されてもよく、コンピューティングデバイス内のVMRトラッカーハードウェア内のプロセッサによって実施されてもよい(たとえば、図7のVMRトラッカーハードウェアブロック706)。

【００６９】

ブロック902において、プロセッサは、コンピューティングデバイス上のプロセスが実行中である現在のVMRに関する情報を記憶し得る。プロセスは、データ、機能、または命令を記憶するメモリの連続部分を表すいくつかのVMRを含むアドレス空間を利用し得る。プロセスのプログラムカウンタは、プロセスが実行されている現在のVMRを指すことができる。記憶される情報は、プロセッサのPID、プロセスのアドレス空間内のすべてのVMRのメモリ範囲、およびプロセスが実行されている現在のVMR(たとえば、プロセスのメインテキスト内の機能)を含み得る。たとえば、この情報は、プロセッサが監視しているプロセスごとに1つずつ、テーブル形式で記憶され得る。情報は、VMRトラッカーハードウェア内のデータストアに記憶され得る。

【００７０】

ブロック904において、プロセッサは、プロセスによって現在実行されている命令に関する新しい情報を受信し得る。たとえば、プロセッサは、時には、プロセスが実行されているCPUから更新されたプログラムカウンタを受信し得る。プロセッサはまた、VMRカーネルドライバからプロセスのVMRのPIDおよび現在のメモリ範囲を受信し得る。

【００７１】

決定ブロック906において、プロセッサは、プロセスが新しい情報に基づいてVMR状態遷移をしていたかどうかを決定し得る。プロセッサは、プロセスが別のVMRに移行したかどうかを決定するために、VMRのプログラムカウンタ、PID、および現在のメモリ範囲を利用し得る。たとえば、プロセスの最後に記憶された現在のVMRがプロセスのメインテキスト内にある場合、プロセッサは、プロセスが同じVMR内でまだ実行中であるか、または別のVMRに遷移したか(たとえば、ライブラリに記憶された関数への関数呼出し)どうかをプログラムカウンタから決定し得る。

【００７２】

プロセスがVMR状態遷移をしていなかったという決定に応答して(決定ブロック906=「No」)、プロセッサは、ブロック904においてプロセスによって現在実行されている命令に関する新しい情報を受信し続けることができる。プロセスは、VMR状態遷移をしていたという決定に応答して(決定ブロック906=「Yes」)、プロセッサは、ブロック908においてVMR状態遷移を記録し得る。たとえば、プロセッサは、プロセスがそこから移行したVMR、およびプロセッサがそこへと遷移したVMRを記憶し得る。次いで、プロセッサは、ブロック902において、新しい現在のVMR情報を記憶し得る。このように、方法900は、プロセスのVMR状態遷移履歴を追跡するためのハードウェアベースの実施形態を提供する。

【００７３】
図10は、様々な実施形態による、コンピューティングデバイス上で実行されるプロセスのVMR状態遷移履歴のソフトウェアベースの追跡のための方法1000を示す。方法1000は、コンピューティングデバイスのプロセッサ(たとえば、図1Aのコンピューティングデバイス100のCPU102)によって実施され得る。
【００７４】
ブロック1002において、プロセッサは、コンピューティングデバイス上のプロセスが実行中である現在のVMRに関する情報を記憶し得る。プロセスは、データ、機能、または命令を記憶するメモリの連続部分を表すいくつかのVMRを含むアドレス空間を利用し得る。プロセスのプログラムカウンタは、プロセスが実行されている現在のVMRを指すことができる。記憶される情報は、プロセッサのPID、プロセスのアドレス空間内のすべてのVMRのメモリ範囲、およびプロセスが実行されている現在のVMR(たとえば、プロセスのメインテキスト内の機能)を含み得る。たとえば、この情報は、プロセッサが監視しているプロセスごとに1つずつ、テーブル形式で記憶され得る。
【００７５】
ブロック1004において、プロセッサは、プロセス上で1つまたは複数のVMR追跡戦略を実施し得る。VMR追跡戦略の例には、選択的追跡(関心のある特定のプロセスのみを追跡する)、周期的追跡(周期的にVMR状態遷移情報を収集する)、日和見的追跡(プロセスが実行されているプロセッサの制御をVMRカーネルドライバが取得したときに、VMR状態遷移情報を収集する)、およびページフォールトに基づく追跡(非現在のすべてのVMRを非実行可能に設定し、VMRの状態遷移が発生したときにページフォールトを追跡する)が含まれ得る。
【００７６】
決定ブロック1006において、プロセッサは、プロセスがVMR状態遷移をしていたかどうかを決定し得る。プロセッサは、1つまたは複数のVMR遷移追跡戦略から、VMR状態遷移が発生したかどうかを決定し得る。たとえば、VMR状態遷移は、プロセスのプログラムカウンタが新しいVMRを指すときに検出されてもよく、ページフォールトに基づく追跡においてページフォールトがあったときに検出されてもよい。
【００７７】
プロセスがVMR状態遷移をしていなかったという決定に応答して(決定ブロック1006=「No」)、プロセッサは、ブロック1004において、プロセスにおいて1つまたは複数のVMR追跡戦略を実施し続けることができる。プロセスは、VMR状態遷移をしていたという決定に応答して(決定ブロック1006=「Yes」)、プロセッサは、ブロック1008においてVMR状態遷移を記録し得る。たとえば、プロセッサは、プロセスがそこから移行したVMR、およびプロセッサがそこへと遷移したVMRを記憶し得る。次いで、プロセッサは、ブロック1002において、新しい現在のVMR情報を記憶し得る。このように、方法1000は、プロセスのVMR状態遷移履歴を追跡するためのソフトウェアベースの実施形態を提供する。
【００７８】
様々な実施形態は、様々なコンピューティングデバイスのいずれかに実施され得、その一例(たとえば、通信デバイス1100)が図11に示されている。様々な実施形態では、通信デバイス1100は、図1Aを参照して本明細書で説明したコンピューティングデバイス100と同様であり得る。このように、通信デバイス1100は、図8～図10の方法800、900、および1000の一部または全部を実施し得る。
【００７９】
通信デバイス1100は、タッチスクリーンコントローラ1104および内部メモリ1106に結合されたプロセッサ1102を含み得る。プロセッサ1102は、一般的または特定の処理タスクのために指定された1つまたは複数のマルチコア集積回路であり得る。内部メモリ1106は、揮発性メモリであっても不揮発性メモリであってもよく、またセキュアおよび/または暗号化メモリ、あるいは非セキュアおよび/または非暗号化メモリ、あるいはそれらの任意の組合せであってもよい。タッチスクリーンコントローラ1104およびプロセッサ1102は、抵抗感知タッチスクリーン、容量感知タッチスクリーン、赤外線感知タッチスクリーンな

どの、タッチスクリーンパネル1112に結合される場合もある。さらに、通信デバイス1100のディスプレイは、タッチスクリーン機能を有する必要はない。

【００８０】

通信デバイス1100は、プロセッサ1102およびアンテナ1110に結合され、セルラー通信を送信および受信するように構成された、セルラーネットワークトランシーバ1108を有し得る。トランシーバ1108およびアンテナ1110は、様々な実施形態の方法を実施するために、本明細書で述べた回路とともに使用され得る。通信デバイス1100は、トランシーバ1108および/またはプロセッサ1102に結合された1つまたは複数のSIMカード1116を含み得、また、本明細書で説明するように構成され得る。通信デバイス1100は、セルラーネットワークを介した通信を可能にし、プロセッサに結合され得るセルラーネットワークワイヤレスモデムチップ1117を含み得る。

【００８１】

通信デバイス1100はまた、オーディオ出力を提供するためのスピーカ1114を含み得る。通信デバイス1100はまた、本明細書で論じる構成要素のすべてまたは一部を収容するための、プラスチック、金属、または材料の組合せから構成されるハウジング1120を含み得る。通信デバイス1100は、使い捨てまたは再充電可能な電池などの、プロセッサ1102に結合された電源1122を含み得る。再充電可能な電池はまた、通信デバイス1100の外部のソースから充電電流を受信するために、周辺デバイス接続ポートに結合され得る。通信デバイス1100はまた、ユーザ入力を受信するための物理的ボタン1124を含み得る。通信デバイス1100はまた、通信デバイス1100をオンまたはオフにするための電源ボタン1126を含み得る。

【００８２】

前述の方法の説明およびプロセスフロー図は単に例示的な例として提供されており、様々な実施形態および実施形態の動作が提示された順序で実行されなければならないことを必要とする、または暗示することが意図されるものではない。当業者によって理解されるように、前述の実施形態および実施形態における動作の順序は、任意の順序で実行され得る。「その後」、「次いで」、「次に」などの単語は、動作の順序を限定することが意図されるものではなく、これらの単語は、単に方法の説明を通じて読者を導くために使用される。さらに、たとえば、冠詞「a」、「an」、または「the」を使用する単数形での請求項要素へのいかなる言及も、要素を単数形に限定するものとして解釈されるべきではない。

【００８３】

本明細書に開示された実施形態および実施形態に関連して説明された様々な例示的な論理ブロック、ユニット、回路、およびアルゴリズム動作は、電子ハードウェア、コンピュータソフトウェア、またはその両方の組合せとして実装され得る。ハードウェアとソフトウェアとのこの互換性を明確に説明するために、様々な例示的な構成要素、ブロック、ユニット、回路、および動作が、それらの機能性に関して一般に本明細書において説明されている。そのような機能性がハードウェアまたはソフトウェアとして実装されるかどうかは、特定のアプリケーションおよびシステム全体に課される設計制約に依存する。当業者は、特定のアプリケーションごとに様々な方法で説明した機能を実装することができるが、そのような実装の決定は、特許請求の範囲から逸脱するものと解釈されるべきではない。

【００８４】

本明細書に開示された実施形態および実施形態に関連して説明された様々な例示的な論理、論理ブロック、ユニット、および回路を実装するために使用されるハードウェアは、様々なプロセッサまたはプロセッサと回路との組合せにおいて実装されてもよく、それらによって実行されてもよい。様々な実施形態を実装することができるプロセッサおよび回路の例は、汎用プロセッサ、デジタル信号プロセッサ(DSP)、特定用途向け集積回路(ASIC)、フィールドプログラマブルゲートアレイ(FPGA)、および他のプログラマブルロジックデバイス、ディスクリートゲートまたはトランジスタロジック、ディスクリートハードウェア構成要素、あるいは本明細書に記載の機能を実行するように設計されたそれらの任意

の組合せを含む。汎用プロセッサは、マイクロプロセッサであってもよいが、代替として、プロセッサは、任意の従来のプロセッサ、コントローラ、マイクロコントローラ、または状態機械であってもよい。プロセッサはまた、コンピューティングデバイスの組合せ、たとえば、DSPとマイクロプロセッサとの組合せ、複数のマイクロプロセッサ、DSPコアと組み合わせた1つまたは複数のマイクロプロセッサ、あるいは他の任意のそのような構成として実装されてもよい。あるいは、いくつかの動作または方法は、所与の機能に特有の回路によって実行され得る。

【００８５】

1つまたは複数の例示的な実施形態および実施形態において、説明された機能は、ハードウェア、ソフトウェア、ファームウェア、またはそれらの任意の組合せにおいて実装され得る。ソフトウェアにおいて実装される場合、機能は、1つまたは複数の命令またはコードとして、非一時的コンピュータ可読記憶媒体または非一時的プロセッサ可読記憶媒体に記憶され得る。本明細書に開示された方法またはアルゴリズムの動作は、非一時的コンピュータ可読記憶媒体またはプロセッサ可読記憶媒体上に存在し得るプロセッサ実行可能ソフトウェアユニットにおいて具体化され得る。非一時的コンピュータ可読記憶媒体またはプロセッサ可読記憶媒体は、命令またはデータ構造の形態で所望のプログラムコードを記憶するために使用され得、またコンピュータまたはプロセッサによってアクセスされ得る任意の記憶媒体であり得る。限定ではなく一例として、そのような非一時的コンピュータ可読記憶媒体またはプロセッサ可読記憶媒体は、RAM、ROM、EEPROM、フラッシュメモリ、CD-ROMまたは他の光ディスク記憶装置、および磁気ディスク記憶装置または他の磁気ストレージデバイスを含み得る。本明細書で使用するディスク(disk)およびディスク(disc)には、コンパクトディスク(CD)、レーザーディスク(登録商標)、光ディスク、デジタル多用途ディスク(DVD)、フロッピーディスク、およびBlu-ray(登録商標)ディスクが含まれ、ディスク(disk)は通常データを磁気的に再生し、ディスク(disc)はデータをレーザーで光学的に再生する。本明細書で説明されるメモリの組合せもまた、非一時的コンピュータ可読媒体およびプロセッサ可読媒体の範囲内に含まれる。さらに、方法またはアルゴリズムの動作は、コンピュータプログラム製品に組み込まれ得る、非一時的プロセッサ可読記憶媒体および/またはコンピュータ可読記憶媒体上のコードおよび/または命令の1つまたは任意の組合せあるいはセットとして存在してもよい。

【００８６】

当業者が特許請求の範囲を作成または使用することを可能にするために、様々な実施形態および実施形態の前述の説明が提供される。これらの実施形態に対する様々な変更は、当業者には容易に明らかであり、本明細書において定義される一般的原理は、特許請求の範囲から逸脱することなしに、いくつかの実施形態に適用され得る。したがって、本開示は、本明細書に示された実施形態および実施形態に限定されることが意図されるものではなく、本明細書に開示された特許請求の範囲、および原理、ならびに新規な特徴と一致する最も広い範囲が与えられるべきである。

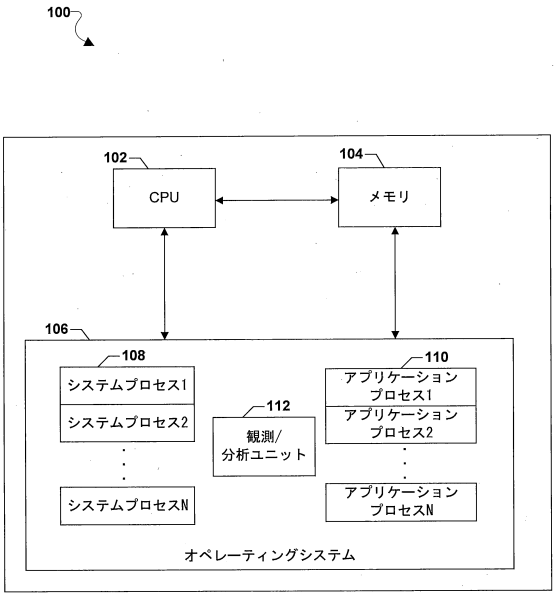
【符号の説明】

【００８７】

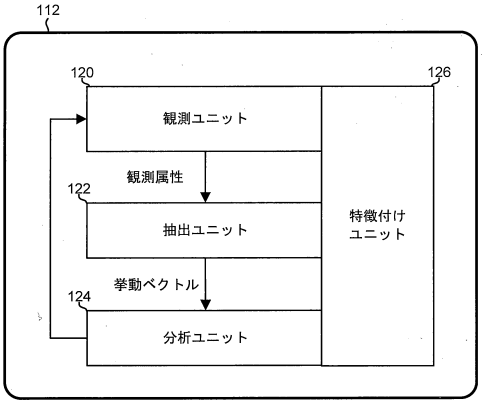
100 コンピューティングデバイス
102 中央処理装置(CPU)
104 メモリ
106 オペレーティングシステム
108 システムプロセス、プロセス
110 アプリケーションプロセス、プロセス
112 観測/分析ユニット
120 観測ユニット
122 抽出ユニット
124 分析ユニット
126 特徴付けユニット

200 プロセスアドレス空間
204 ユーザスタック
206 共有ライブラリ
208 libcライブラリ
208a VMR
210 ヒープ
212 メインテキスト
214 プログラムカウンタ(PC)
302 スタック
304 フロー図
312 スタック
314 フロー図
402 ヒープ
404 フロー図
412 ヒープ
414 フロー図
502 スタック
504 フロー図
600 VMR監視システム
602 監視される構造的属性
604 監視される構造的属性
606 監視される構造的属性
608 監視される構造的属性
610 監視される構造的属性
614 出力
700 システム
702 CPUコア
704 VMRカーネルドライバ
706 VMRトラッカーハードウェアブロック
708 データバッファ
710 VMRテーブル
712 VMR遷移ロガー
800 方法
900 方法
1000 方法
1100 通信デバイス
1102 プロセッサ
1104 タッチスクリーンコントローラ
1106 内部メモリ
1108 セルラーネットワークトランシーバ、トランシーバ
1110 アンテナ
1112 タッチスクリーンパネル
1114 スピーカ
1116 SIMカード
1117 セルラーネットワークワイヤレスモデムチップ
1120 ハウジング
1122 電源
1124 物理的ボタン
1126 電源ボタン

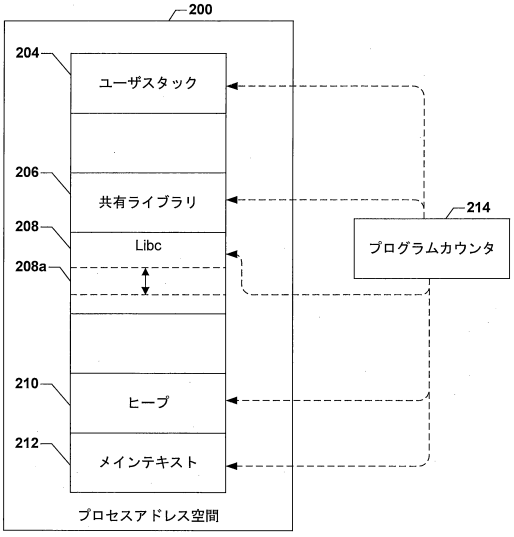
【図１Ａ】



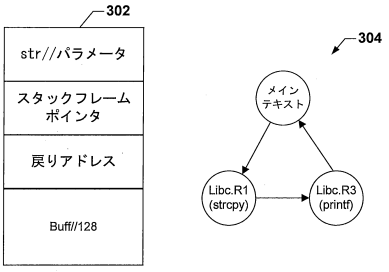
【図１Ｂ】



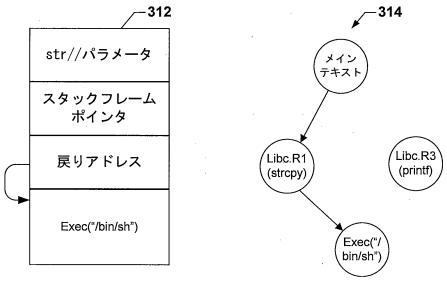
【図２】



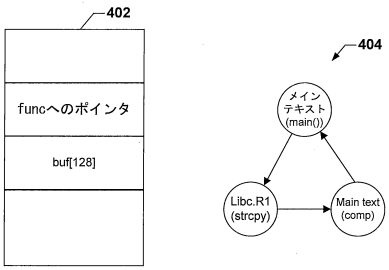
【図３Ａ】



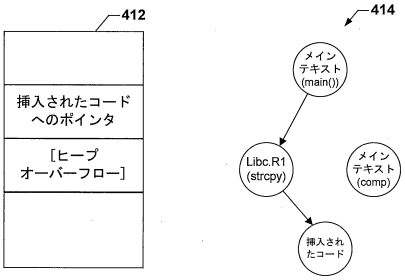
【図３Ｂ】



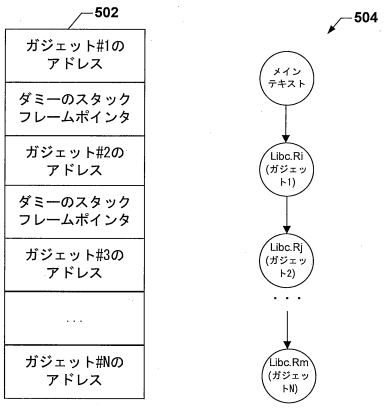
【図４Ａ】



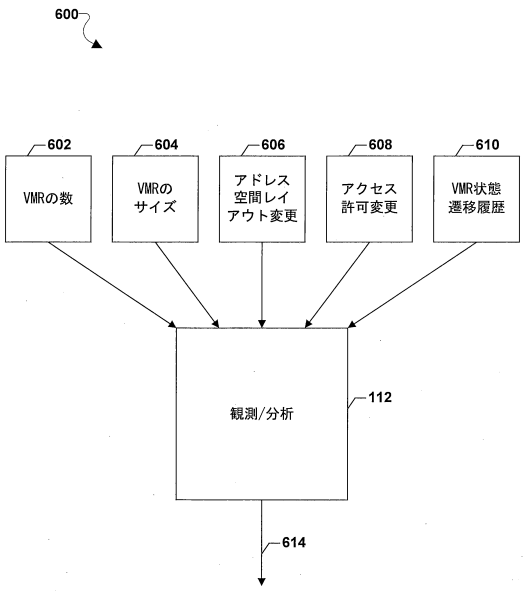
【図４Ｂ】



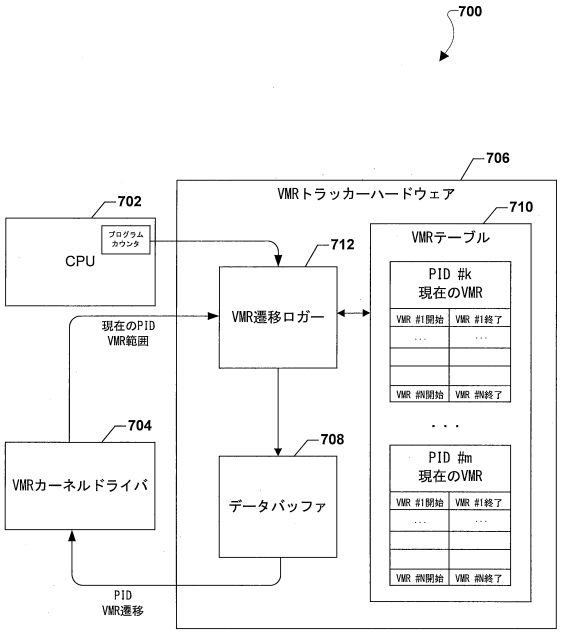
【図５】



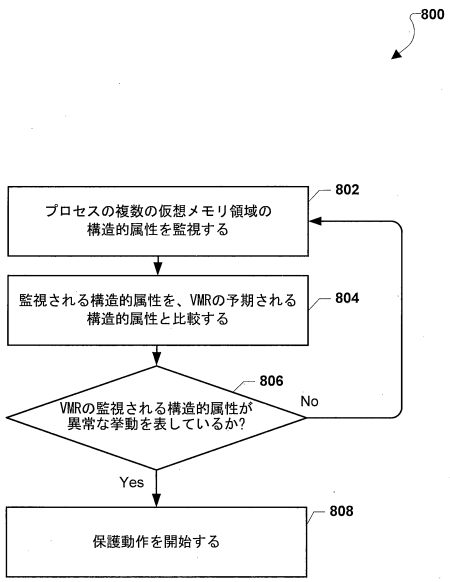
【図６】



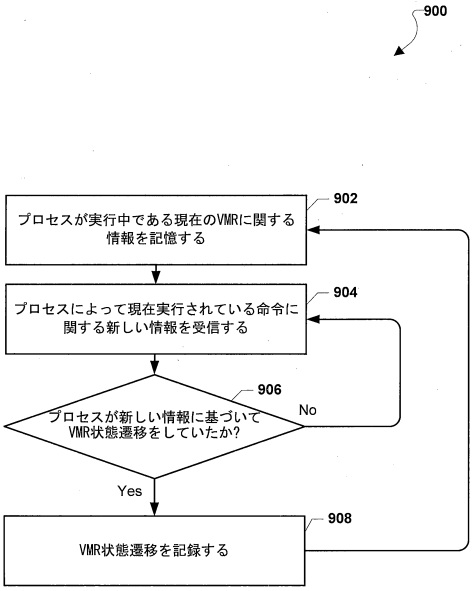
【図７】



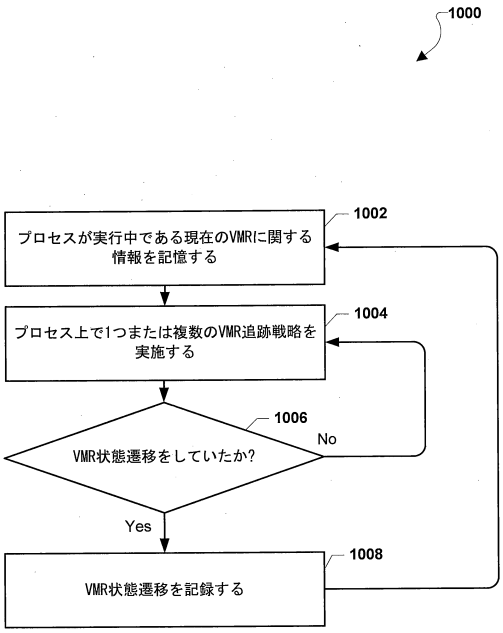
【図８】



【図９】



【図１０】



【図１１】

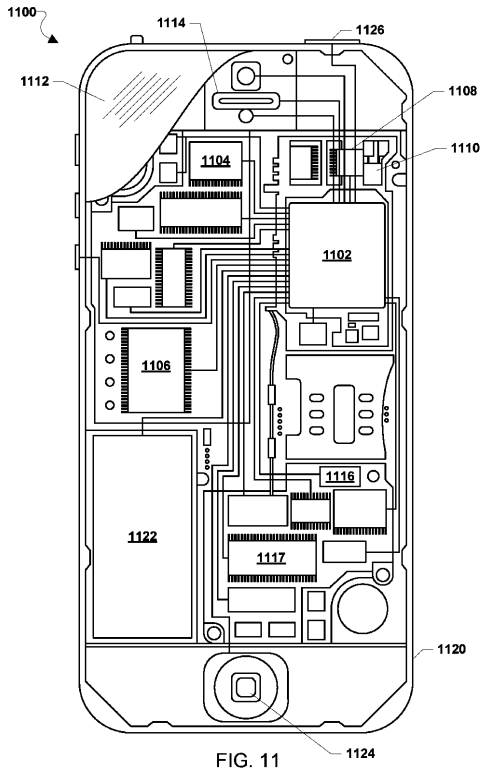


FIG. 11

フロントページの続き

(72)発明者　ラジャルシ・グプタ
アメリカ合衆国・カリフォルニア・９４０８７・サニーベール・カーライル・ウェイ・７２９
(72)発明者　ナイーム・イスラム
アメリカ合衆国・カリフォルニア・９２１２１－１７１４・サン・ディエゴ・モアハウス・ドライヴ・５７７５

審査官　平井　誠

(56)参考文献　特開２００６－１７２００３（ＪＰ，Ａ）
特開２０１０－０９２１７４（ＪＰ，Ａ）
米国特許出願公開第２０１１／０１７９４９０（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　２１／５５－５６
Ｇ０６Ｆ　１２／１４