

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第6400794号
(P6400794)

(45) 発行日 平成30年10月3日 (2018. 10. 3)

(24) 登録日 平成30年9月14日 (2018. 9. 14)

(51) Int. Cl.	F I
G 0 6 F 21/56 (2013. 01)	G O 6 F 21/56 3 6 0
G 0 6 F 9/455 (2006. 01)	G O 6 F 9/455
G 0 6 F 9/445 (2018. 01)	G O 6 F 9/445

請求項の数 20 外国語出願 (全 16 頁)

(21) 出願番号	特願2017-126083 (P2017-126083)	(73) 特許権者	515348585
(22) 出願日	平成29年6月28日 (2017. 6. 28)		エーオー カスペルスキー ラボ
(65) 公開番号	特開2018-97841 (P2018-97841A)		AO Kaspersky Lab
(43) 公開日	平成30年6月21日 (2018. 6. 21)		ロシア国, 125212 モスクワ, レニ
審査請求日	平成29年7月28日 (2017. 7. 28)		ングラドスコ ショス 39エー/3
(31) 優先権主張番号	2016148489		39A/3 Leningradskoe
(32) 優先日	平成28年12月12日 (2016. 12. 12)		Shosse, Moscow 125
(33) 優先権主張国	ロシア (RU)		212, Russia
(31) 優先権主張番号	15/446, 157	(74) 代理人	110001139
(32) 優先日	平成29年3月1日 (2017. 3. 1)		S K 特許業務法人
(33) 優先権主張国	米国 (US)	(74) 代理人	100130328
			弁理士 奥野 彰彦
		(74) 代理人	100130672
			弁理士 伊藤 寛之

最終頁に続く

(54) 【発明の名称】 インタプリタによるコード実行のシステム及び方法

(57) 【特許請求の範囲】

【請求項 1】

インタプリタによるプログラムコード実行の方法であって、工程 A～E を含み、
前記工程 A では、エミュレートされたコンピュータ環境においてインタプリタがプログラムコードの命令を実行し、

前記工程 B では、前記インタプリタに解釈規則の無い未知のオブジェクトに関連する前記プログラムコードの命令を前記インタプリタが検出したときに、前記インタプリタは前記プログラムコードの命令のさらなる実行を停止し、

前記工程 C では、前記未知のオブジェクトの実行結果に対応する実行結果を有する補助コードであって、インタプリタが解釈規則を有する既知のオブジェクトを含むものを前記
インタプリタが取得し、

前記工程 D では、前記インタプリタが前記補助コードの命令を実行し、

前記工程 E では、前記補助コードの実行が完了した後に、前記インタプリタが前記プログラムコードの命令の実行を再開する、

方法。

【請求項 2】

前記オブジェクトは、プロシージャ、クラスのインスタンス、クラスのインスタンスのメソッド又は変数、スクリプト、ファイル、補助コードのうちの1つである、請求項1に記載の方法。

【請求項 3】

10

20

前記補助コードは、前記エミュレートされたコンピュータ環境から取得される、請求項1に記載の方法。

【請求項4】

前記補助コードは、ローカルデータベース又はリモートデータベースから取得される、請求項1に記載の方法。

【請求項5】

前記インタプリタの実行環境は、前記補助コードのために初期化される、請求項1に記載の方法。

【請求項6】

前記プログラムコードの実行及び前記補助コードの実行は、前記プログラムコードの字句解析の実行、前記プログラムコードの構文解析の実行、中間コードの生成、解釈規則に従った中間コードの解釈、のうちの1つ以上を含む、請求項1に記載の方法。

10

【請求項7】

前記補助コードは、他の補助コードが取得される少なくとも1つの他の未知のオブジェクトをさらに含み、前記他の補助コードの実行結果は、前記他の未知のオブジェクトの実行結果に対応する、請求項1に記載の方法。

【請求項8】

前記解釈されたプログラムコードのアンチウィルスチェックを実行する工程をさらに含む、請求項1に記載の方法。

【請求項9】

20

インタプリタによりプログラムコードを実行するシステムであって、エミュレートされたコンピュータ環境においてインタプリタを実行するハードウェアプロセッサを含み、

前記インタプリタは、

前記プログラムコードの命令を実行し、

前記インタプリタに解釈規則の無い未知のオブジェクトに関連する前記プログラムコードの命令を前記インタプリタが検出したときに、前記インタプリタは前記プログラムコードの命令のさらなる実行を停止し、

前記未知のオブジェクトの実行結果に対応する実行結果を有する補助コードであってインタプリタが解釈規則を有する既知のオブジェクトを含むものを前記インタプリタが取得し、

30

前記インタプリタが前記補助コードの命令を実行し、

前記補助コードの実行が完了した後に、前記インタプリタが前記プログラムコードの命令の実行を再開するように構成されているシステム。

【請求項10】

前記オブジェクトは、プロシージャ、クラスのインスタンス、クラスのインスタンスのメソッド又は変数、スクリプト、ファイル、補助コードのうちの1つである、請求項9に記載のシステム。

【請求項11】

前記補助コードは、前記エミュレートされたコンピュータ環境から取得される、請求項9に記載のシステム。

40

【請求項12】

前記補助コードは、ローカルデータベース又はリモートデータベースから取得される、請求項9に記載のシステム。

【請求項13】

前記インタプリタの実行環境は、前記補助コードに対して初期化される、請求項9に記載のシステム。

【請求項14】

前記プログラムコードの実行及び前記補助コードの実行は、前記プログラムコードの字句解析の実行、前記プログラムコードの構文解析の実行、中間コードの生成、解釈規則に従った中間コードの解釈、のうちの1つ以上を含む、請求項9に記載のシステム。

50

【請求項 15】

前記補助コードは、他の補助コードが取得される少なくとも1つの他の未知のオブジェクトをさらに含み、前記他の補助コードの実行結果は、前記他の未知のオブジェクトの実行結果に対応する、請求項9に記載のシステム。

【請求項 16】

前記解釈されたプログラムコードのアンチウィルスチェックを実行する工程をさらに含む、請求項9に記載のシステム。

【請求項 17】

インタプリタによってプログラムコードを実行するためのコンピュータ実行可能命令を格納する非一過性の記録媒体であって、命令A～Eを含み、

10

前記命令Aは、エミュレートされたコンピュータ環境においてインタプリタがプログラムコードの命令を実行させ、

前記命令Bは、前記インタプリタに解釈規則の無い未知のオブジェクトに関連する前記プログラムコードの命令を前記インタプリタが検出したときに、前記インタプリタが前記プログラムコードの命令のさらなる実行を停止させ、

前記命令Cでは、前記未知のオブジェクトの実行結果に対応する実行結果を有する補助コードであってインタプリタが解釈規則を有する既知のオブジェクトを含むものを前記インタプリタが取得させ、

前記命令Dでは、前記インタプリタが前記補助コードの命令を実行させ、

前記命令Eでは、前記補助コードの実行が完了した後に、前記インタプリタが前記プログラムコードの命令の実行を再開させる、

20

記録媒体。

【請求項 18】

前記補助コードは、前記エミュレートされたコンピュータ環境から取得される、請求項17に記載の記録媒体。

【請求項 19】

前記補助コードは、ローカルデータベース又はリモートデータベースから取得される、請求項17に記載の記録媒体。

【請求項 20】

前記プログラムコードの実行及び前記補助コードの実行は、前記プログラムコードの字句解析の実行、前記プログラムコードの構文解析の実行、中間コードの生成、解釈規則に従った中間コードの解釈、のうちの1つ以上を含む、請求項17に記載の記録媒体。

30

【発明の詳細な説明】

【技術分野】

【0001】

本発明は概してインタプリタの分野に関し、より具体的にはインタプリタによるプログラムコード実行のためのシステム及び方法に関する。

【背景技術】

【0002】

コンピュータプログラムコードに悪意のある機能が存在するか分析するために、アンチウィルスアプリケーションによってエミュレータが用いられることが増えている。そのような方法は、通常、スクリプトプログラミング言語で書かれたコードを分析するために使用される。一般に、エミュレータは構文解析器（パーサ）とインタプリタを含む。構文解析器は、分析されるプログラムコードを中間コード（バイトコードなど）に変換する。インタプリタは、中間コード及びエミュレートされた実行環境を使用して、解釈規則に従って中間コードの命令を連続的に実行（すなわち、解釈）する。エミュレーション中、分析されるコードの実行はエミュレートされた環境でなされるため、分析されるコードが実際のコンピュータシステムのリソースにアクセスすることはできない。従って、エミュレータで実行されているコードは、コンピュータ上のファイルを削除又は変更したり、ファイルからデータを読み込んだり、実行されているコンピュータシステムに危害を加えたりす

40

50

ることはできない。

【0003】

同時に、実行中のコード全てのオブジェクト（関数、プロシージャ及びメソッド、クラスのインスタンスなど）を解釈できるエミュレータを実装することは非常に困難である。サイバー犯罪者はこの事実を利用し、ほとんど使用されないオブジェクトをプログラムコードに含めることが多い。従って、特定のオブジェクト（関数など）の戻り値がこのオブジェクトの実行の期待値と一致しない場合、悪意のあるコードはエミュレータ内で自身が実行されていることを「理解」し、更なる実行を終了し、又は、悪意のある機能を実行せず、検出されないようにする。

【0004】

反対に、アンチウィルスアプリケーションのメーカーは、サイバー犯罪者が悪意のあるコードに含ませる新たなオブジェクトについて、解釈規則をエミュレータへ効率的に追加しようと努力している。しかしながら、これは通常、エミュレータ自体のプログラムコードの変更を要する。このような変更のたびに、エミュレータのプログラムコードのテストとデバッグが必要となり、エミュレータの更新版をアンチウィルスアプリケーションのユーザに効率的に提供することができない。ここで、インタプリタの実際のプログラムコードを変更することなく、調査対象のプログラムコードの命令であってインタプリタに解釈規則（プログラムコードの実行結果が、対応する命令の実行結果に対応するもの）の無いオブジェクトを含むものを解釈（すなわち、インタプリタによる実行）しなければならない、という技術的問題が発生する。オブジェクトは、プロシージャ、クラスのインスタンス、クラスのインスタンスのメソッド又は変数、スクリプト又はファイルであってもよい。

【0005】

しかしながら、既存のアンチウィルス技術では、悪意のあるコードであってエミュレータ（より具体的にはそのインタプリタ）に解釈規則の無いオブジェクトが含まれているものを検出することができないため、前述の技術的問題を解決することができない。

【発明の概要】

【0006】

本発明は、解釈規則の無いプログラムコードをインタプリタが実行するためのシステム、方法及びコンピュータプログラム製品に関する。本発明の技術的結果の1つは、補助プログラムコードを解釈することにより、インタプリタ自体を変更することなく、調査対象プログラムコードであって解釈規則の無いものをインタプリタが実行する品質を改善することにある。

【0007】

1つの例示的な態様では、インタプリタによるプログラムコード実行の方法であって、工程A～Eを含み、上記工程Aでは、エミュレートされたコンピュータ環境においてインタプリタがプログラムコードの命令を実行し、上記工程Bでは、上記インタプリタに解釈規則の無い未知のオブジェクトに関連する上記プログラムコードの命令を上記インタプリタが検出したときに、上記インタプリタは上記プログラムコードの命令のさらなる実行を停止し、上記工程Cでは、上記未知のオブジェクトの実行結果に対応する実行結果を有する補助コードであって、インタプリタが解釈規則を有する既知のオブジェクトを含むものを上記インタプリタが取得し、上記工程Dでは、上記インタプリタが上記補助コードの命令を実行し、上記工程Eでは、上記補助コードの実行が完了した後に、上記インタプリタが上記プログラムコードの命令の実行を再開する。

【0008】

1つの例示的な態様では、上記オブジェクトは、プロシージャ、クラスのインスタンス、クラスのインスタンスのメソッド又は変数、スクリプト、ファイル、補助コードのうちの1つである。

【0009】

1つの例示的な態様では、上記補助コードは、上記エミュレートされたコンピュータ環

10

20

30

40

50

境から取得される。

【0010】

1つの例示的な態様では、上記補助コードは、ローカルデータベース又はリモートデータベースから取得される。

【0011】

1つの例示的な態様では、上記インタプリタの実行環境は、上記補助コードのために初期化される。

【0012】

1つの例示的な態様では、上記プログラムコードの実行及び上記補助コードの実行は、上記プログラムコードの字句解析の実行、上記プログラムコードの構文解析の実行、中間コードの生成、解釈規則に従った中間コードの解釈、のうちの1つ以上を含む。

10

【0013】

1つの例示的な態様では、上記補助コードは、他の補助コードが取得される少なくとも1つの他の未知のオブジェクトをさらに含み、上記他の補助コードの実行結果は、上記他の未知のオブジェクトの実行結果に対応する。

【0014】

1つの例示的な態様では、上記解釈されたプログラムコードのアンチウィルスチェックを実行する工程をさらに含む。

【0015】

1つの例示的な態様では、インタプリタによりプログラムコードを実行するシステムであって、エミュレートされたコンピュータ環境においてインタプリタを実行するハードウェアプロセッサを含み、上記インタプリタは、上記プログラムコードの命令を実行し、上記インタプリタに解釈規則の無い未知のオブジェクトに関連する上記プログラムコードの命令を上記インタプリタが検出したときに、上記インタプリタは上記プログラムコードの命令のさらなる実行を停止し、上記未知のオブジェクトの実行結果に対応する実行結果を有する補助コードであってインタプリタが解釈規則を有する既知のオブジェクトを含むものを上記インタプリタが取得し、上記インタプリタが上記補助コードの命令を実行し、上記補助コードの実行が完了した後に、前記インタプリタが前記プログラムコードの命令の実行を再開するように構成されているシステム。

20

【0016】

1つの例示的な態様では、インタプリタによってプログラムコードを実行するためのコンピュータ実行可能命令を格納する非一過性の記録媒体であって、命令A～Eを含み、上記命令Aは、エミュレートされたコンピュータ環境においてインタプリタがプログラムコードの命令を実行させ、上記命令Bは、上記インタプリタに解釈規則の無い未知のオブジェクトに関連する上記プログラムコードの命令を上記インタプリタが検出したときに、上記インタプリタが上記プログラムコードの命令のさらなる実行を停止させ、上記命令Cでは、上記未知のオブジェクトの実行結果に対応する実行結果を有する補助コードであってインタプリタが解釈規則を有する既知のオブジェクトを含むものを上記インタプリタが取得させ、上記命令Dでは、上記インタプリタが上記補助コードの命令を実行させ、上記命令Eでは、上記補助コードの実行が完了した後に、上記インタプリタが上記プログラムコードの命令の実行を再開させる。

30

40

【0017】

本発明の例示的な態様の上記簡略化された概要は、本発明の基本的な理解を促すためのものである。この概要は、全ての企図された態様の広範な概要ではなく、また、全ての態様の重要若しくは決定的な要素を特定したり、本発明の任意の又は全ての態様の範囲を描写したりすることも意図していない。前述の目的を達成するために本発明の1つ又は複数の側面が記載され、これは特に特許請求の範囲において主張される特徴を含むものである。

【図面の簡単な説明】

【0018】

50

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、本開示の1つ以上の例示的な態様を示し、詳細な説明とともに、それらの原理及び実施形態を説明する役割を果たす。

【図1】図1は、インタプリタによるプログラムコード実行のための例示的なシステムを示す。

【図2】図2は、インタプリタによるプログラムコード実行のための例示的なシステムを示す。

【図3】図3は、インタプリタによるプログラムコード実行のための例示的な方法を示す。

【図4】図4は、インタプリタによるプログラムコード実行のためのシステム及び方法の態様を実施することができる汎用コンピュータシステムの一例を示す。

【詳細な説明】

【0019】

ここで説明される本発明の例示的な態様は、インタプリタによるプログラムコード実行のためのシステム、方法、及びコンピュータプログラム製品に関する。当業者であれば、以下の説明は例示的なものに過ぎず、決して限定するものではないことを理解するであろう。他の態様は、本開示の恩恵を受ける当業者には容易に示唆されるであろう。以下、添付の図面に例示されている例示的な態様の実装を詳細に参照する。図面及び以下の記載の全体に渡って、同一の参照符号が可能な範囲で使用され、同一又はそれに相当するものについては、同一の参照符号を用いるものとする。

【0020】

図1は、本発明の一態様によるインタプリタによるプログラムコード実行のための例示的なシステムを示す。アンチウィルスアプリケーション101は、それが展開されているコンピュータシステム上に情報セキュリティを提供するコンピュータプログラムである。アンチウィルスアプリケーション101は、解析のためにプログラムコード102をエミュレータ200に渡す。1つの例示的な態様では、エミュレータ200は、アンチウィルスアプリケーション101の一部である。ソースプログラムコード102は、JavaScript、VBScript、Python等のスクリプト言語で記述され得る。1つの例示的な態様では、エミュレータ200は、これらに限定はしないが、字句解析器103、構文解析器104、意味解析器105、中間コード生成器106、インタプリタ107、ロギングモジュール115、及び分析モジュール113を含み得る。1つの例示的な態様では、インタプリタ107は仮想マシンを含む。

【0021】

1つの例示的な態様では、字句解析器103は、ソースプログラムコード102を構成する文字列を読み取り、これらの文字を意味のある列又は語彙にグループ化する。各字句について出力トークンが構築され、構文解析器104に渡される。構文解析器104は、トークンを使用してトークンの文字列の文法構造又は構文木を記述するツリー状の中間コードを生成する。意味解析器105は、上記構文木を使用して、言語定義との意味的整合性についてソースプログラムコードをチェックする。中間コード生成器106は中間コードを生成し、それはインタプリタ107に渡される。中間コードは統一され、ソースコード102の記述言語に依存しない。

【0022】

インタプリタ107は、実行環境110内のコードを実行（又は解釈）する。実行環境110（ランタイム環境）は、実行されているコードの名前付きオブジェクトに対する割り当てスキーム及びメモリの問題を解決し、変数へのアクセス方法、プロシージャ間のリンク、パラメータ転送の仕組み、オペレーティングシステム・I/Oデバイス・他のプログラムとの相互作用、を提供する。結果的に、実行環境110の初期化は、メモリの割り当てと、変数及びデータへのアクセスをインタプリタ107に提供することを含む。インタプリタ107は、中間コード生成器106によって作成された中間コードのみを実行することができる点に注目すべきである。説明の便宜のために、調査対象コードと中間コードという用語は同義であるとみなす。中間コード又は調査対象コードは、機械に依存しないコードである（異な

るコンピュータアーキテクチャにおいてインタプリタによって実行される)。従って、ソースコード102が記述されたプログラミング言語にかかわらず、字句、構文及び意味解析器103-105及び中間コード生成器106の働きにより、統一された言語文法から調査対象コードが作成され、インタプリタ107はこれを実行することができる。

【0023】

ローカルデータベース111は、実行環境110にリンクされ、調査対象コードのオブジェクトの解釈規則を含む。1つの例示的な態様では、オブジェクトは、プロシージャ、クラスのインスタンス、クラスのインスタンスのメソッド又は変数のうちの1つである。さらに別の例示的な態様では、調査対象コードは、埋め込まれたスクリプトコードセクション109を含むことができ、同様にオブジェクトであるファイル108を扱うためのコードも含むことができる。この場合、インタプリタ107によるコードの実行は再帰的に行われる。例えば、埋め込まれたスクリプトは、中間コードに変換されていないので、インタプリタ107によって実行することができない。従って、インタプリタ107によって実行される中間コードを中間コード生成器106で生成するために、埋め込まれたスクリプト109を字句解析器103にもう一度送らなければならない。

【0024】

調査対象コードの実行中に、ファイルがディスク108に保存された場合、このファイルの内容はインタプリタ107によって実行されてもよい。しかし、ファイル内容には変換されていないコードが含まれているため、ファイルはアンチウィルス101に引き渡され、その後、モジュール103-106によって中間コードへ変換される。

【0025】

本発明は、コンピュータセキュリティの分野及びプログラムコードのアンチウィルススキャンのためのインタプリタの使用に限定されないことに留意されたい。しかしながら、本発明は、インタプリタによるコード実行が用いられる他のコンピュータ技術分野に適用できることも同様に理解されるべきであり、例えば、コードインタプリタにおける解釈規則を更新する方法を単純化することに用いられることも考えられる。

【0026】

1つの例示的な態様では、インタプリタ107をロギングモジュール115へ更にリンクすることができる。当該モジュール115は、インタプリタ107における中間コードの各命令の実行中に、インタプリタのログ114へ記録を行うことができる。

【0027】

1つの例示的な態様では、分析モジュール113は、アンチウィルスルールに従ってアンチウィルスレコードデータベース112の記録と一致するようにインタプリタのログ114の記録をチェックするように構成される。アンチウィルスルールは、調査対象ファイルが悪意のあるもの、又は悪意のあるものではないと認識される条件を指定する。もし前述の記録が一致すれば、調査対象コードは悪意のあるものと判断され、分析モジュール113はこの情報をインタプリタ107に渡す。この場合、インタプリタ107はコードのさらなる実行を中止し、制御をアンチウィルス101に渡す。アンチウィルス101は検出された脅威（悪意のあるファイル）をコンピュータユーザーに警告し、調査対象コードの悪意のある内容を排除するための適切な措置を講じる。

【0028】

以下の表1は、インタプリタのログ114とアンチウィルスレコードデータベース112で同時に生じるアンチウィルスレコードの例を示す。通常、アンチウィルスレコードは、シグネチャと呼ばれる。表の最初のレコードにはレコードオープンが含まれている。これは、URLアドレス「http://***** exe ***.jpg」（"*"は任意の文字を意味する）にあるファイルを取得する要求がされたことを意味する。2番目のレコードは、上記要求がURLアドレスに送られたことを意味する。3番目のレコードは、ファイル"c:\\temp*****.exe"へのデータ保存を意味する。4番目のレコードは、保存されたファイルの実行を開始することを意味する。これら4つの連続した動作の実行は、通常、悪意のあるファイルの特徴であることが明らかである。従って、この例によるアンチウィルスルールは

次のようになる。即ち、インタプリタのログ114に表1のレコード1～4のシーケンスが含まれている場合、調査対象コードには「HEUR:Trojan-Downloader.Script.Generic:（表2参照）の判定が与えられる。この判定は、調査対象コードとソースコード102が悪質なファイル、トロイの木馬のダウンローダ、であることを意味する。

【表 1】

No.	レコード	
	名称	内容
1	オープン	msxml2.xmlhttp.open(“get”, “http://*****exe**. jpg”)
2	送信	msxml2.xmlhttp.send(“[get,http://*****exe**. jpg]”)
3	保存	adodb.stream.savetofile(“c:¥¥temp¥¥*****. exe” , “2”)
4	実行	wscript.shell.run(“c:¥¥temp¥¥*****. exe” , “0” , “0”)

10

【表 2】

アンチウィルスルール
<pre>\$on (オープン -> 送信 -> 保存 -> 実行) { \$判定(“HEUR:Trojan-Downloader.Script.Generic”); }</pre>

20

【0 0 2 9】

図2は、本発明の一態様によるインタプリタによるプログラムコード実行のための例示的なシステムを示す。インタプリタ107は、調査対象コード（中間コード）201の各命令を連続して実行するために使用される。インタプリタ107は、インタプリタに解釈規則の無いオブジェクト（すなわち、未知のオブジェクト）を含む調査対象コード201の命令実行へと移ると、調査対象コード201のさらなる実行が中止される。解釈規則とはプログラムコードであって、その実行結果が対応する命令の実行結果に一致するものをいう。次に、インタプリタ107は、その実行結果が未知のオブジェクトの実行結果に一致する補助コードを取得し、それらオブジェクト（未知のオブジェクト）を含む補助コードについては、インタプリタ107が解釈規則を有する。そして、インタプリタは補助コードの各命令を連続して実行し、補助コードの実行が完了した後に、調査対象コード201の次の命令から実行を再開し、最後の命令が実行されるまで調査対象コード201を実行する。

30

【0 0 3 0】

1つの例示的な態様によれば、オブジェクトは、プロシージャ、クラスのインスタンス、クラスのインスタンスのメソッド又は変数、スクリプト、ファイルのうちの1つである。

40

【0 0 3 1】

図3は、インタプリタによるプログラムコード実行のための例示的な方法を示す。調査対象コード201は、ステップ301において、インタプリタ107の入力に到達し、インタプリタ107はステップ202において調査対象コード201の各命令を連続して実行する。インタプリタ107に解釈規則の無いオブジェクト（未知のオブジェクト）を含む調査対象コード201の命令を実行し始めた場合（ステップ303における検証）、ステップ304において調査対象コードの更なる実行を停止し、ステップ305において、その実行結果が前述の未知のオブジェクトの実行結果に一致する補助コード306をインタプリタが取得する。補助コード306

50

には、インタプリタ107内に解釈規則が存在するオブジェクト（未知のオブジェクト）のみが含まれている。次に補助コード306がステップ301のインタプリタ107に渡され、そこで補助コード306の各命令を連続して実行し、補助コードの実行を完了した後、ステップ301の調査対象コードの次の命令から実行を再開する。補助コード306の実行完了の検証は、ステップ302及び307で行われる。本発明の方法の動作は、調査対象コード201の最後の命令が実行される瞬間まで続く。この状態はステップ302及び307で検証される。調査対象コード201の最後の命令を実行した後、ステップ308でインタプリタ107の動作が終了する。

【0032】

1つの例示的な態様では、ステップ301において、インタプリタ107は、インタプリタ107の実行環境110を初期化する（ステップ203）。さらに別の例示的な態様では、補助コード306はインタプリタ107の実行環境110に含まれ、補助コード306はステップ305において実行環境110から取得される。

【0033】

1つの例示的な態様では、補助コード306は、ローカルデータベース111から取得される。この場合、補助コード306の実行環境（ステップ203）は初期化され得る。なぜなら、インタプリタ107の実行環境110初期化時においては、補助コード306によって使用されるオブジェクト及び変数を含まないからである。

【0034】

1つの例示的な態様では、補助コード306がローカルデータベース111に存在しない場合、補助コード306はリモートデータベース（図示せず）から取得される。

【0035】

1つの例示的な態様では、ロギングモジュール115を使用して、インタプリタのログ114に、実行中のコードの各命令又は命令の一部を記録する（ステップ301）。インタプリタのログ114に記録を作成するために必要な命令のリストは、エミュレータ200に含まれていてもよい。解析モジュール113は、前述の記録を作成する際に、アンチウィルスルールに従って、インタプリタ107のログ内の記録とアンチウィルスレコードデータベース112の整合性を確認する。もし前述の記録が一致する場合、調査対象コード102は悪意のあるものと判断される。

【0036】

1つの例示的な態様では、コードの実行及び補助コードの実行は、字句解析器103を用いた字句解析、構文分析器104を用いた構文分析、意味解析器105を用いた意味解析、中間コード生成器106による中間コードの生成、インタプリタ107による解釈規則に従った中間コードの解釈、のステップを含む。

【0037】

補助コード306は、ソースコード102の言語又は変換された調査対象コード201の言語、すなわち変換された形式、で記述され得ることに留意されたい。例えば、ソース言語がJavaScriptの場合、変換された言語はJava仮想マシンのバイトコードとなる。1つの例示的な態様では、補助コード306がソースコード102の言語で書かれているとき、補助コード306は、まず字句分析器103に渡され、次に構文解析器104及び意味解析器105に引き渡され、最後に中間コード生成器106は補助コード306から変換された形式の補助コードを構築し、そしてインタプリタ107によって実行される。例示的な一態様では、補助コード306の変換の後、変換されていない補助コード306と共に、又は別個の記憶/キャッシュに格納される。この補助コードが次にインタプリタ107によって要求されると、変換された形式の補助コードがインタプリタ107に返される。

【0038】

別の例示的な態様では、補助コード306が変換された形式で最初に提供されたとき、補助コード306はインタプリタ107に一度に送信されて実行される。

【0039】

以下の表3は、悪意のあるソースコード107の一部の例を示す。理解を深めるために述べ

10

20

30

40

50

ると、変換された調査対象コード201ではなく、実際のソースコード102が与えられる。行4は、解釈規則が当初欠けていたオブジェクト「Scripting.Dictionary」の呼び出しを含む。この例では、オブジェクト「Scripting.Dictionary」は不明なオブジェクトである。従って、ステップ4においては、インタプリタ107で調査対象コードの実行が停止され、インタプリタ107は、表4から補助コード306を提供され、その実行結果は、表3の4行目のオブジェクト「Scripting.Dictionary」を呼び出す際の実行結果に対応している。補助コード306を実行した後、調査対象コードの実行は表3の5行目以降に再開される。表4の補助コード306は、解釈規則がインタプリタ107内に存在するオブジェクト（未知のオブジェクト）のみを含む。

【0040】

ファンクションといった稀なオブジェクトの呼び出しは、通常悪意のあるファイルの特徴であることに注意すべきである。このようにして、悪意のあるファイルは、実際のコンピュータではなくエミュレータで実行されているかどうかをチェックする。

【表3】

行	ソースコード
1	var yoryymvz=3;
2	var izzfhpg=jiiyamm.split("");
3	var rtrslnmj="";
4	var y=new ActiveXObject("Scripting.Dictionary");
5	y.add("a","t");
6	if (y.Item("a")== "t")
7	{
8	rtrslnmj="bjd";
9	}
10	else
11	{
12	yoryymvz=0;
13	}
14	;
15	for (i=0; i<izzfhpg.length; i+=yoryymvz)
16	{
17	bsvuyeigy=bsvuyeigy+izzfhpg[i];
18	}
19	var rtrslnmj="bjdebjdvbjdabjdl".split(rtrslnmj).join("");
20	this[rtrslnmj](bsvuyeigy);

10

20

30

40

【表 4】

行	補助コード
1	obj = {};
2	obj.prototype.add = function(p1, p2)
3	{
4	this[p1] = p2;
5	}
6	obj.prototype.Item = function(p)
7	{
8	return this[p];
9	}
10	Scripting.Dictionary = obj;

10

【0041】

さらに別の例示的な態様では、補助コード306は、他の未知のオブジェクトをさらに含むことができる。この他の未知のオブジェクトは対応する他の補助コードを得ることができ、その実行結果は当該他の未知のオブジェクトの実行結果に対応する。

20

【0042】

表5は、既知のオブジェクトの例を示す。表6は、未知のオブジェクトの補助コードがどのようにして既知及び未知のオブジェクトを含むのか、その例を示す。

【0043】

例えば、未知のオブジェクトGの補助コードには、既知のオブジェクトA、B、Cのみが含まれている。同時に、未知のオブジェクトHの補助コードには、既知のオブジェクトA、B及び他の未知のオブジェクトGの両方が含まれている。しかし、当該他の未知のオブジェクトGについては反対に、既知のオブジェクトA、B、Cに依存する別の補助コードも存在する。

30

【0044】

このように、上に挙げた例では、未知のオブジェクトHを含んだ調査対象コード201内の命令がステップ202において実行されると、当該調査対象コードの実行はステップ304で停止される。次に、ステップ202で実行が再開されるオブジェクトHの補助コード306がステップ305で取得される。オブジェクトHの補助コードが実行される時に、もし他の未知のオブジェクトGを含む命令が実行された場合、ステップ305において当該他の未知のオブジェクトGについて異なる補助コードが新たに取得され、その実行はステップ202で再開される。オブジェクトGの補助コードの実行を完了した後、オブジェクトHの補助コード306の実行が再開され、オブジェクトHの補助コード306の実行を完了した後、調査対象コード201の実行が再開される。言うまでもなく、調査対象コード201は再度補助コードを取得する必要がある他の未知のオブジェクトを含み得る。さらに、既知のオブジェクトに対する未知のオブジェクトの依存性は、必要に応じて深く埋め込まれ得る。

40

【表 5】

既知のオブジェクト
A, B, C, D, E, F

【表 6】

未知のオブジェクト	オブジェクトを含む補助コード
G	A, B, C
H	A, B, G

【0045】

インタプリタによる調査対象プログラムコード実行の質の向上からなる、上述した本発明のシステム及び方法の1つの技術的結果は、補助プログラムコードの解釈によって成し
 10 遂げられる。インタプリタによる調査対象プログラムコード実行の質の向上は、調査対象
 プログラムコードがインタプリタに解釈規則の無いオブジェクトを含む場合であっても、
 インタプリタが調査対象プログラムコードを実行し続けることにある。従来技術の既知の
 インタプリタ及びエミュレータは、未知の機能を含む命令を実行するときにはエラーを伴
 って実行を完了する。

【0046】

インタプリタに変更を加えることなく、インタプリタに解釈規則の無い調査対象プロ
 グラムコードの命令についてインタプリタによる実行を可能にすることからなる、上述した
 本発明のシステム及び方法の別の技術的結果は、補助プログラムコードの解釈によって成
 し遂げられる。この技術的な結果は、インタプリタに解釈規則の無い命令について当該解
 20 釈規則が無いことを理由に調査対象コードの実行は停止されず、インタプリタによって補
 助コードが実行されるため成し遂げられる。

【0047】

図4は、インタプリタによるプログラムコード実行のためのシステム及び方法の態様に
 従って実装され得る例示的な汎用コンピュータシステムである。図示のように、コンピュ
 ータシステム20（パーソナルコンピュータまたはサーバでもよい）は、中央処理装置21、
 システムメモリ22及び中央処理装置21に関連するメモリを含む様々なシステム構成要素に
 接続されるシステムバス23を含む。当業者には理解されるように、システムバス23は、
 バスメモリまたはバスメモリコントローラ、周辺機器用バス、及び他のバスアーキテク
 チャと相互作用することができるローカルバスを含むことができる。システムメモリは、パ
 ーマネントメモリ（ROM）24及びランダムアクセスメモリ（RAM）25を含むことができる。
 30 基本入出力システム（BIOS）26は、ROM24を利用してオペレーティングシステムをロード
 するときのような、コンピュータシステム20の要素間で情報を転送するための基本的な手
 順を格納することができる。

【0048】

コンピュータシステム20はまた、データを読み書きするためのハードディスク27と、リ
 ムーバブル磁気ディスク29に読み書きするための磁気ディスクドライブ28と、CD-ROM、DV
 D-ROM、及び他の光メディアといったリムーバブル光ディスク31を読み書きするための光
 学ドライブ30とを含むことができる。ハードディスク27、磁気ディスクドライブ28、及び
 光学ドライブ30はそれぞれ、ハードディスクインターフェース32、磁気ディスクインター
 フェース33及び光学ドライブインターフェース34を介してシステムバス23に接続されてい
 40 る。ドライブ及び対応するコンピュータ情報メディアは、コンピュータ命令、データ構造、
 プログラムモジュール、及びコンピュータシステム20の他のデータを格納するための電
 源独立モジュールである。

【0049】

例示的な態様では、コントローラ55を介してシステムバス23に接続されたハードディス
 ク27、リムーバブル磁気ディスク29及びリムーバブル光ディスク31を利用するシステムを
 含む。コンピュータによって読み取り可能な形式で保存することができる如何なるタイ
 プのメディア56（ソリッド・ステート・ドライブ、フラッシュ・メモリ・カード、デジタル
 ・ディスク、ランダムアクセスメモリ（RAM）、その他）も利用することが当業者によ
 50 って理解される。

【0050】

コンピュータシステム20は、オペレーティングシステム35が格納されるファイルシステム36と、追加のプログラムアプリケーション37と、他のプログラムモジュール38と、プログラムデータ39とを有する。コンピュータシステム20のユーザは、キーボード40、マウス42、または当業者に知られている任意の他の入力デバイス、例えばマイクロホン、ジョイスティック、ゲームコントローラ、スキャナだがこれらに限らない、を利用してコマンド及び情報を入力することができる。そのような入力デバイスは、典型的には、シリアルポート46を介してコンピュータシステム20に接続され、シリアルポート46は、システムバスに接続されるが、入力デバイスを他の方法、例えば、パラレルポート、ゲームポート、またはユニバーサルシリアルバス（USB）だがこれらに限定されない、を介して接続することもできることが当業者であれば理解できる。モニタ47または他のタイプのディスプレイ装置も、ビデオアダプタ48などのインタフェースを介してシステムバス23に接続することができる。モニタ47に加えて、パーソナルコンピュータは、ラウドスピーカ、プリンタ等その他の周辺出力装置（図示せず）を備えることもできる。

10

【0051】

コンピュータシステム20は、1つまたは複数のリモートコンピュータ49へのネットワーク接続を利用して、ネットワーク環境で動作することができる。上記1つのリモートコンピュータ（または複数のリモートコンピュータ）49は、コンピュータシステム20の性質を説明する上で前述した要素のほとんどまたはすべてを含むローカルコンピュータワークステーションまたはサーバであってもよい。ルータ、ネットワークステーション、ピアデバイス、または他のネットワークノードなどの他のデバイスも、コンピュータネットワークに存在することができるが、これに限定されるものではない。

20

【0052】

ネットワーク接続は、ローカルエリアコンピュータネットワーク（LAN）50及びワイドエリアコンピュータネットワーク（WAN）を形成することができる。このようなネットワークは、企業のコンピュータネットワーク及び社内ネットワークで利用され、一般的にインターネットへアクセスできる。LANまたはWANネットワークでは、パーソナルコンピュータ20は、ネットワークアダプタまたはネットワークインタフェース51を介してローカルエリアネットワーク50に接続される。ネットワークが利用される場合、コンピュータシステム20は、モデム54、またはインターネットなどのワイドエリアコンピュータネットワークとの通信を可能にする当業者に周知の他のモジュールを利用することができる。モデム54は、内部または外部の装置であってもよく、シリアルポート46によってシステムバス23に接続されてもよい。上記ネットワーク接続は、通信モジュールを利用して1つのコンピュータによって別のコンピュータへの接続を確立する多くの周知の方法の非限定的な例であることは、当業者には理解されるであろう。

30

【0053】

様々な態様において、本明細書に記載のシステム及び方法は、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み合わせで実施されてもよい。ソフトウェアで実施される場合、これらの方法は、非一過性のコンピュータ可読媒体上に1つ以上の命令またはコードとして格納されてもよい。コンピュータ可読媒体は、データストレージを含む。限定ではない例として、そのようなコンピュータ可読媒体は、RAM、ROM、EEPROM、CD-ROM、フラッシュメモリ、若しくは他のタイプの電気、磁気、または光学記憶媒体、又は命令若しくはデータ構造の形態で所望のプログラムコードを実行または格納するために利用することができ、汎用コンピュータのプロセッサによってアクセスすることができる他の媒体を含む。

40

【0054】

様々な態様において、本開示で説明されたシステム及び方法は、モジュールに関して扱うことができる。本明細書で使用される「モジュール」という用語は、実世界のデバイス、コンポーネント、又は、例えば、特定用途向け集積回路（ASIC）またはフィールドプログラマブルゲートアレイ（FPGA）等の、ハードウェアを利用して実装されたコンポーネン

50

トの配置、若しくは、（実行されている間に）マイクロプロセッサシステムを特殊用途の装置に変換するモジュールの機能を実装するためのマイクロプロセッサシステム及び命令の集合によるものといった、ハードウェアとソフトウェアの組み合わせ、のことを示す。モジュールは、これら2つの組み合わせとして実装されてもよく、特定の機能はハードウェア単体によって促進され、他の機能はハードウェアとソフトウェアの組み合わせによって促進されてもよい。特定の実施例では、モジュールの少なくとも一部、場合によっては全てが、汎用コンピュータのプロセッサ上で実行されてもよい（例えば、上記図4に詳細に記載されたものなど）。従って、各モジュールは、様々な適切な構成で実現されてもよく、ここに例示される特定の实装に限定されるべきではない。

【0055】

10

明確化のために述べるが、本態様の一連の特徴の全てが本明細書に開示されるわけではない。本開示のいかなる実装の開発においても、開発者の特定の目標を達成するために多数の実装に係る特有の決定がなされなければならない、これらの具体的な目標は異なる実装及び異なる開発者によって変わるであろう。そのような開発努力は複雑で時間がかかるが、それにもかかわらず、この開示の恩恵を受ける当業者にとって通常の工学的な取り組みであることが理解される。

【0056】

更に、本明細書で使用される表現または用語は、限定ではなく説明のためのものであり、本明細書に開示される教示及び指針に照らして、関連技術における当業者の知識と組み合わせ、これら表現または用語が当業者によって解釈されると理解されたい。更に、本明細書または特許請求の範囲における任意の用語が、明示的に記載されていない限り、一般的でないまたは特別な意味で解釈されることを意図するものではない。

20

【0057】

本明細書で開示される様々な態様は、説明のために本明細書で言及される既知のモジュールに対する現在及び将来の既知の等価物を包含する。更に、態様及び用途が示され、記載されているが、本開示の利益を有する当業者には、本明細書に開示された発明概念から逸脱することなく上述したより多くの改変が可能であることが明らかであろう。

【図 1】

1/4

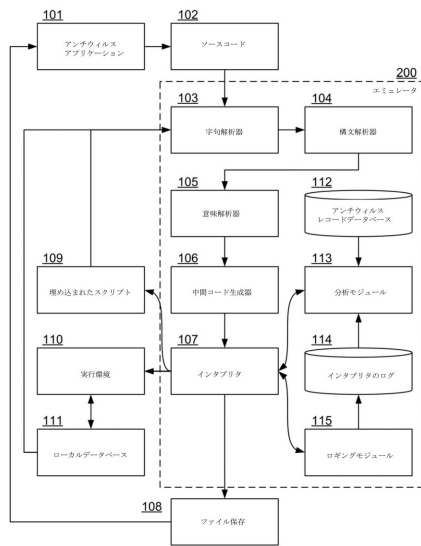


Fig. 1

【図 2】

2/4

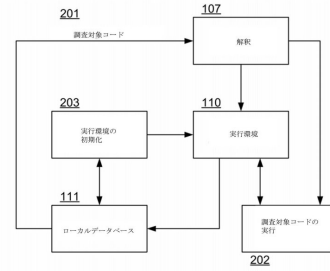


Fig. 2

【図 3】

3/4

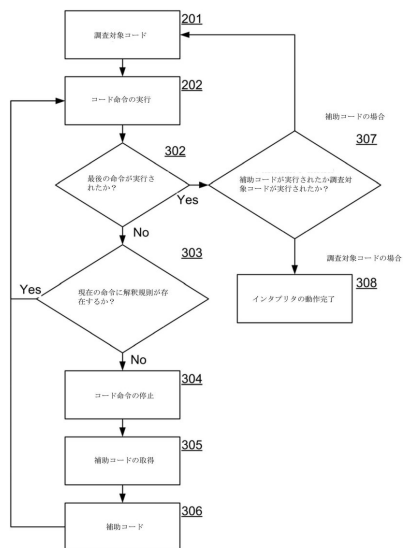


Fig. 3

【図 4】

4/4

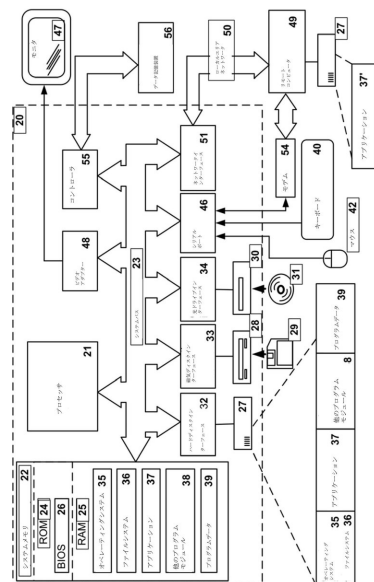


Fig. 4

フロントページの続き

- (72)発明者 ヴァシリー エー. ダヴィドフ
ロシア 125212 モスクワ, レニングラドスコ ショス 39エー/3, エーオー カ
スペルスキー ラボ
- (72)発明者 ドミトリー ブイ. ヴィノグラドフ
ロシア 125212 モスクワ, レニングラドスコ ショス 39エー/3, エーオー カ
スペルスキー ラボ
- (72)発明者 ロマン ワイ. ガブリルチェンコ
ロシア 125212 モスクワ, レニングラドスコ ショス 39エー/3, エーオー カ
スペルスキー ラボ
- (72)発明者 ドミトリー エー. キルサノフ
ロシア 125212 モスクワ, レニングラドスコ ショス 39エー/3, エーオー カ
スペルスキー ラボ

審査官 金沢 史明

- (56)参考文献 特表2009-541898 (JP, A)
特開2008-27306 (JP, A)

- (58)調査した分野(Int.Cl., DB名)
G06F 21/56
G06F 9/455
G06F 9/445