科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　１２　日現在

機関番号：　８２６２６
研究種目：　挑戦的萌芽研究
研究期間：　2012～2013
課題番号：　２４６５００３２

研究課題名（和文）コンピューターウィルスの進化を分析する手法の研究

研究課題名（英文）A Method for Analyzing Evolution of Computer Viruses

研究代表者
森　彰（MORI, Akira）

独立行政法人産業技術総合研究所・知能システム研究部門・研究グループ長

研究者番号：３０３１１６８２

交付決定額（研究期間全体）：（直接経費）　　2,700,000円、（間接経費）　　810,000円

研究成果の概要（和文）：コンピューターウィルスなどの悪意ある攻撃プログラムがどのように変異・進化しているかを解析するための自動化技術の開発を行った。まず、実行プログラムのバイナリーコードを解析して得られる制御フローグラフの間の類似度をもとに進化系統樹の推定を行い、これらが概ね正しい結果を示すことを実際のサンプルを用いて確認した。次に、大局的な分析を行うために、典型的な攻撃パターンの同定を行い、これらのパターンが変異・進化の過程でどのような現れ方をするかを分析した。この過程で、対象がバイナリーコードであっても、関数や手続きのように共用して用いられる文脈依存コードを同定して解析することができる新たな手法を開発した。

研究成果の概要（英文）：We have developed an automated method for analyzing evolution processes of computer viruses. We successfully reconstructed phylogenetic trees for real-world computer virus samples by comparing control flow graphs obtained by binary code analysis of executable programs. We also examined how frequent attack patterns of computer viruses appear along estimated evolution processes. In doing so, we developed a new method for identifying and analyzing context dependent shared code segments, such as functions and procedures, in binary executables without assuming the use of high-level programming languages.

研究分野：　総合領域

科研費の分科・細目：　情報学、計算機システム・ネットワーク

キーワード：　ネットワークセキュリティ技術　コンピューターウィルス　進化予測　バイナリーコード解析

１．研究開始当初の背景

コンピューターウィルスによる攻撃が爆発的に増大しており、これらを人間の手作業で解析することは不可能になってきている。しかし、コンピューターウィルスの攻撃は、特定の状況にならないと発動しないものがほとんどであり、その動作を観察しているだけでは攻撃の全容を知ることができない。また、例外なく難読化が施されており、リターン命令や間接ジャンプ命令の飛び先アドレスを計算しない従来のツール（逆アセンブラ等）による解析は困難であった。このように、バイナリコードを静的に解析する手法がウィルス対策として重要であるが、コンピューターウィルスのようにコンパイラーから生成されているとは限らないプログラムに対して有効なツールの開発は存在しないのが現状であった。

２．研究の目的

年を追って巧妙化と複雑化が進むコンピューターウィルスを、仮想コンピューター上でのエミュレーションと間接ジャンプアドレスの記号評価による静的解析を組み合わせることで自動解析し、結果として得られる制御フローグラフの類似度により分類・検索を行ったり、分子系統学アルゴリズムの適用により変異の状況を系統樹として視覚的に表示したりする技術を確立することを主たる目的とする。そして、特徴的な攻撃パターンを網羅的に細分化し類型化しておき、その共起関係や時系列の傾向を分析することで、特定のサンプル群が今後どのような機能を備えて進化していくかを予測する技術についての研究も行う。実際のウィルスサンプルを対象にした実験を行い、手法の有効性を確認するとともに、実用化へ向けた検討も進める。

３．研究の方法

これまでの研究により、機械命令を静的単一代入形式と呼ばれる中間表現に変換しながら、飛び先アドレスを記号的に計算して、制御フローグラフを順次拡張していく展開型の静的解析と、仮想 PC 上での実行による動的解析を組み合わせて、暗号化が施された実際のウィルスのサンプルに対しも、その振舞いの多くを自動解析することが可能になっている。本研究では、解析の結果得られるものは、API 関数呼び出しの情報が付与された制御フローグラフである。これらのグラフの類似度のマトリクスを計算すれば、自動的にクラスタリングを行い、類似度による検索を行うことが可能になる。類似度のマトリクスに対し分子系統学のアルゴリズムを適用すれば、その進化過程を推定し系統樹として可視化することもできる。本研究ではまず、開発したツールを並列処理により高速化し、数千個レベルのウィルスに対しても、こうした処理が実用時間で行えるようにする。解析の精度が低ければ類似度から計算される進化系統樹の信頼度が低下するため、ウィルス攻撃に特有の、プロセス乗っ取りや自己コード注入といったパターンの有無をもとにした進化系統樹の推定と照合しながら、解析ツールの性能を向上させ、最終的に実際のウィルスのサンプルに対して、妥当な進化系統樹が導出できるようにする。そして、ウィルス攻撃の詳細な類型化を行い、サンプル群の質的な分類や進化傾向の予測を行う手法を確立する。

４．研究成果

(1)初年度は、これまでに開発を行ってきたバイナリコード解析ツールを用いて、多数のコンピューターウィルスの解析を行うことができるように処理性能を向上させる試みを行った。具体的には、独立して処理することが可能な間接ジャンプ命令の飛び先アドレスの計算を、それまでに作成された静的単一代入形式の制御フロー図をコピーすることで、複数プロセスで行うように静的解析ツールに変更を加えた。実験の結果、ループを含むコード片のように、複雑な飛び先アドレスが複数ある場合は並列化の効果が見られた。しかし、並列計算のメリットがあまり大きくない場合も確認されたため、計算効率の向上の方法として、Python の JIT コンパイラー処理系である PyPy を利用することを試みた。結果として、ツールの改変とくに繰り返し計算部の調整を行ったところ、計算速度が平均して 3 倍程度速くなることが確認できた。さらに、計算結果のキャッシュ効率の向上を試み、キャッシュされた結果を保持する条件を精査してアルゴリズムの改善を行ったところ、約 1.5 倍程度の速度向上が見られ、全体として 4 倍程度の速度向上を達成することができ、実用へ向けて大きく前進することができた。

次に、改善されたツールをハードウェア仮想化ベースのエミュレーターと組み合わせて、実際のウィルスサンプルの自動解析実験を行った。ツール等の不具合により部分的な制御フロー図しか得られない場合もあったが、100 個を超えるサンプルに対してほぼ完全な制御フロー図を自動生成することができた。この際、近年多く見られる、複数のプロセスやスレッドにまたがって攻撃を行うウィルスを自動解析するためのエミュレーターの改変を行う必要が生じた。

当初の想定ではウィルスプログラムの実行により起動されるプロセスのみを解析の対象としており、ウィルスプログラムが起動する子プロセスや独立して動く別プロセスに注入される遠隔スレッドのコードの解析は手作業で行うことを想定していた。しかし、近年のコンピューターウィルスでは、自分自身コピーして子プロセスとして起動したり、別プロセスのアクセス権限を修正して、自分自身のコードの部分を当該プロセスのメモ

リ空間に書き出して遠隔スレッドとして起動したりする場合が多くあり、人手による作業は複雑になり、解析の効率の面で大きな問題であった。

こうした複数のプロセスやスレッドにまたがった**多次元攻撃**を自動解析するには、解析タスクをプロセス・スレッドごとに多重化する必要があり、これを効率よく自動化することは簡単ではない。仮想実行による動的解析においても、全ての実行ログを保存しておくことはできないため、実行時になんからの多重化を行うことが必要となるが、これをコンピューターウィルスに検知されずに行うことは簡単ではない。本研究では、ハイパーバイザーモニターを利用して、プロセスおよびスレッドの切り替えを検知し、静的解析を多重化する方法を提案し実装した。

最終的に、コンピューターウィルスの進化過程を解析する実験を行い、実行されるウィルスプログラムのメインスレッドから開始して、制御フロー図を支配木で近似して木構造の類似度計算を行い、クラスタリング、検索、進化系統樹推定の実験を行った。目視により、おおむね妥当な結果が得られていることを確認することができた。亜種の存在の可視化などもできており、実用上の価値は大きいと思われる。

(2)次年度では、バイナリコード解析において、複数箇所から呼び出される関数等の文脈依存共有コードを解析する手法を提案し、その実装を行った。具体的には、解析に用いる静的単一代入形式を拡張し、文脈に応じて複数の値を持つ飛び先のデータについて、その呼び出し文脈を指定する疑似関数を導入する手法を考案し、値の解析アルゴリズムを実装した。この解析手法を適用したところ、複雑なコンピューターウィルスのプログラムであってもその制御フローを高い精度で解析できることが確認できた。攻撃パターンの推定については、二次攻撃プログラム投入、アンチウィルス機能の無力化、システム常駐、ボットネットワーク活動、外部記憶デバイス感染、などの十数個の典型パターンの分類と定義を行い、推定された進化系統樹の上での位置づけを確認した。当初は、個別の攻撃パターンを単位として、変異や進化の過程が切り出せると期待していたが、実験を行った範囲ではそうした現象は特定できなかった。より粒度の細かいパターンの同定が必要かどうかの検討を継続して行っているところである。

一方で、コンピューターウィルスのプログラムの解析においても、通常のプログラムにおける関数や手続きなどのように、複数箇所から呼び出されて複数箇所に復帰する制御フローに依存したプログラム構造をまたいでデータフローの解析を行う必要があることが明らかになってきた。プログラム言語で書かれたソースコードを対象として、静的解析手法の一つの問題として研究されてきたが、バイナリコードを対象とした研究はこれまで行われてこなかった。バイナリコードでは、コンパイラーで作成される通常のプログラムのように、スタックと CALL/RET 命令を用いて関数の機能を実装するという制約はなく、文脈依存の共有コードを実現する方法は無数にあるため、一般的な解析手法を新たに確立する必要が生じた。

具体的には、共用コードに相当する制御フローを複写するのではなく、共用コードへ進入するフローと退出するフローを対応させるための疑似関数を導入して SSA 形式を拡張した Fibered SSA(F-SSA)形式と呼ばれる手法を提案し実装した。図１に通常の関数を解析した場合に得られる SSA 形式の制御フロー図を示す。ここで、関数のリターンアドレスに相当する変数 $dest_1$ を評価するとその値は $\Phi_w(v{:}0xBBBB, v'{:}0xCCCC)$ となる。これは、先行ノード v と v' を持つノード w において導入された Φ関数によりマージされた値を表し、リターンアドレスが文脈に応じて複数あることを示している。本研究では、このように Φ関数によって表されたアドレスが飛び先に現れた場合には、文脈依存の共用コードが存在すると仮定し、それを**手続き**と呼ぶ。すなわち、ノード w を入り口とした手続きがあり、呼び出しがノード v の場合にはその帰り先はアドレス 0xBBBB から始まるノードとなり、また呼び出しがノード v' の場合には帰り先は 0xCCCC になると考える。出口ノードは $dest_1$ が現れるノードになる。

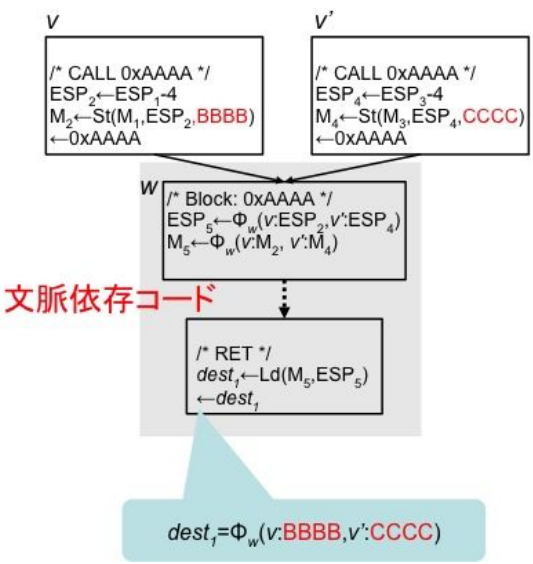

図１　文脈依存コードの例

この前提のもとに、どのように文脈に依存した制御フローを識別して解析を行うかを考える。以下、図２に示したコードをもとに説明する。この例では、復帰アドレスを EBX に格納した上で、EAX に格納された共用コードを呼び出している。共用コードは 0x401014 にある XOR ECX,ECX だけであり、これを実行

した後に EBX に格納されているアドレスに復帰する。共用コードは 0x40100a と 0x401011 から二回呼び出されている。

```
start:
            mov   eax,   offset
proc_B
        mov ebx, offset ret1
        jmp eax
  ret1:
        mov ebx, offset ret2
        jmp eax
  ret2:
        ret

  proc_B proc
        xor ecx, ecx
        jmp ebx
  proc_B endp
```

図２　サンプルコード

従来の手法では、手続きの範囲、特に出口ノードを同定することが求められ、これを見逃した場合に無駄なコピーを作成してしまうという問題があった。また、出口ノードをより広く補足しようとした場合、手続きの入れ子構造によってはデッドロックを引き起こす場合が生じ、ヒューリスティクな手法で回避するしかなくなるという問題が生じた。今回提案した Fibered SSA (F-SSA)形式では、こうした問題を回避するために、そもそもコピーを行うのではなく、手続きへの進入パスと退出パスを対応づけて値の評価が行えるようにする手法を提案した。これは従来の SSA 形式に、Φ関数の引数を取り出す作用をもつ疑似関数である Π関数を追加して、一つのノードからフロー依存の複数のエッジが張られることを許すものである。Φ関数により分岐する文脈依存のアドレスに相当する式 $\Phi_w(v_0{:}e_0,\ldots,v_n{:}e_n)$があった場合にそれに対応して双対な疑似関数 $\Pi_{vi\to w}(X)(0\le i\le n)$ を挿入することで、SSA 形式の性質を保ったまま、複数の進入パスと退出パスの対応付けを行うことが可能になる。

図 3 に図 2 のコードを解析した制御フロー図の概形を示す。ここでは、共用コードであるノード 3 がコピーされておらず、代わりにノード 3 への進入パスでラベル付けされたエッジが張られている。ノード 3 を呼び出すのはノード 2 とノード 4 であるので、2→3 と 4→3 というラベルを貼られたノード 3 からの退出パスが明示されている。このため、あたかもループがあるように見えても実際にループはなく、複数の文脈が制御フロー図に畳み込まれている状態となっている。

この状況で、値の計算が対応して行われるために、

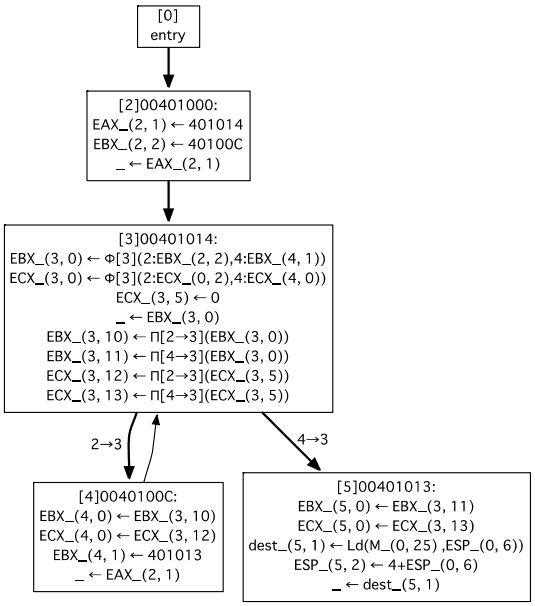

図３　F-SSA 形式の解析例

Φ関数と Π関数には射影規則と呼ばれる以下の代数規則が成り立つ必要がある。

$$\Pi_{vi\to w}(\Phi_w(\ldots,v_i{:}e_i,\ldots s))\Rightarrow e_i$$

これを利用することで、後続のノード 4 やノード 5 において、文脈を取り違えることなく値の評価ができることになる。

F-SSA 形式による解析は、制御フローの複写を行うよりも効率がよく、生成される制御フロー図も簡潔な場合が多い。ただし、Π関数が含まれる式の評価において、対応するΦ関数を考慮する必要があったり、Π関数の引数に現れる変数の出現に依存した処理、たとえばループ展開などについて特別な注意が必要になったりするなど、より高度な処理を要する場合もあることが分かっている。

５．主な発表論文等

〔学会発表〕（計２件）

①森 彰、泉田 大宗、バイナリコードの全プログラム解析手法について、2014 年 暗号と情報セキュリティシンポジウム（SCIS2014）

②森 彰、泉田 大宗、複数のプロセスとスレッドにまたがるマルウェア攻撃の自動解析について、2013 年 暗号と情報セキュリティシンポジウム（SCIS2013）

６．研究組織

(1)研究代表者

森 彰（MORI, Akira）

産業技術総合研究所・知能システム研究部門・研究グループ長

研究者番号：３０３１１６８２