「マルチメディア，分散，協調とモバイル
(DICOMO2018)シンポジウム」 平成30年7月

制御機器ペネトレーションテスト支援のための ソフトウェア状態遷移推定方式の提案

木藤 圭亮[†1] 西川 弘毅[†1] 山本 匠[†1] 河内 清人[†1]

概要：制御機器などへのサイバー攻撃が後を絶たないなか、製品やシステムの出荷前に脆弱性が残存していないかチェックするペネトレーションテストが重要となっている。しかし一般にペネトレーションテストは、専門的な知識や技能を有するペンテスターと呼ばれる者が、豊富な経験と知識を総動員してシステムの内部処理を推理しながら行うため、非効率的であり、その結果はペンテスターの技量に強く依存する。本研究では、制御システムに対するペネトレーションテストの支援方式として、制御機器の入出力から相関値を算出し、相関値の時間変化が大きいタイミングを、システムの状態遷移として判定する方式を提案する。本方式を利用することで、システムの状態遷移に着目したペネトレーションテストの支援を行う事ができ、ペネトレーションテストの属人性を廃することが期待できる。

Software State Transition Estimation Method for Supporting Penetration Test of Networked Control Systems

KEISUKE KITO[†1] HIROKI NISHIKAWA[†1]
TAKUMI YAMAMOTO[†1] KIYOTO KAWAUCHI[†1]

1. はじめに

IoT 機器や制御機器を狙ったサイバー攻撃が近年頻発している。制御システムを狙った攻撃事例としては、PLC を狙った Stuxnet 事件[1]や、車載機器を狙った Jeep Hack[2]の事例が記憶に新しい。

制御システムに対するサイバー攻撃が多発するなか、機器やシステムの出荷前に脆弱性が残存していないかどうかを試験する、ペネトレーションテストが重要となっている。ペネトレーションテストとは既に知られているサイバー攻撃技術を用いて、実際にシステムへ侵入を試みることで、脆弱性がないかどうかを検査する手法である[1]。情報系システムに対しては、Web システムなどをはじめとしてペネトレーションテストが広く実施されてきた。制御システムについては、IPA の制御システムのセキュリティリスク分析ガイド[5]において、脆弱性検査やペネトレーションテストを行い、制御システムに対するサイバー攻撃のリスクを確認、低減することが推奨されている。

一方でペネトレーションテストは、豊富な経験や知識を有する、ペンテスターと呼ばれる専門家によって実施される。これは一般的なソフトウェア試験などとは異なり、システムの仕様が分からないブラックボックスな状態でテストを行うためである。ペンテスターは、攻撃データなどを入力したときのシステムの出力や応答、挙動などから、内部処理やデータ構造などを推定し、脆弱性を発生させる入力を経験などから導き出す。そのためペネトレーションテストすべてを自動化することは難しい。

一方でペネトレーションテストをはじめとするセキュリティ検査では、試験するコードカバレッジが高いほど多くの脆弱性を発見できることが、経験則的に示されている[3]。そのため、セキュリティ検査では、多くの状態遷移を行わせて、ペネトレーションテスト中のコードカバレッジを向上させることが、ペネトレーションテストの品質向上につながる。

そこで本研究では制御機器を対象とした、システムの挙動を表す状態遷移を推定するペネトレーションテストの支援方式を提案する。制御機器は一般に、入力と何らかの関連のある出力を行う。そのため入力と出力の間で相関が起きやすい。この相関の状態をシステムの一つの状態としてとらえ、入出力の相関値が時間的に大きく変化したときに、状態変化が発生したと判断する。適当な入力に対する出力を観測し、相関値算出を行い、時系列変化が一定以上の時に、状態変化が発生したと判断し、逐次状態遷移を記録することで、制御機器の状態遷移を自動的に推定することが可能となる。

本提案方式を利用することで、ブラックボックスな制御機器に対しても、提案方式で推定した状態遷移の情報を用いることで、ペネトレーションテストを効率的に行う事ができ、ペネトレーションテストのシステムの状態遷移推定についての属人性を廃することが期待でき、品質の安定したペネトレーションテストが実現可能になる。

†1 三菱電機株式会社 情報技術総合研究所

2. 制御機器とペネトレーションテスト

制御システムへのサイバー攻撃事例が後を絶たない。近年では2010年のStuxnet事件や、2015年に発表されたJeep Hackなどが記憶に新しい。

Stuxnet事件は、2010年にイランの核施設においてSiemens社製のPLCのラダープログラムが不正に書き換えられ、制御する遠心分離機を破壊した。これを契機に制御システムのセキュリティが叫ばれるようになった。

2015年のBlack Hat USAにてCharlie Millerらは、Jeep社のCherokeeという車種において、インターネット経由で攻撃が可能で、車載ECUのファームウェアなどを不正に書き換えて、遠隔で操舵できてしまう脆弱性を発表した。

これらの2つの攻撃事例をきっかけに、制御システムに対する情報セキュリティが重要視されるようになった。IPAの制御システムのセキュリティリスク分析ガイド[5]では、制御システムが各社独自のシステムから、標準化・共通化が行われ、サイバー攻撃の標的になりやすいことが示されている。さらに制御システムに対する脆弱性検査やペネトレーションテストを行い、制御システムに対するサイバー攻撃のリスクを確認、低減することが推奨されている。つまり、情報系システムだけではなく、制御システムに対してもペネトレーションテストなどのセキュリティ試験を行うことが必要とされている。

ペネトレーションテスト(Penetration Test、以下、ペンテスト)とは、ネットワーク接続されたシステムに対して、既知の攻撃技術を用いて実際に侵入を試みることで、システムに脆弱性がないかどうかを試験する手法である。侵入テストなどとも呼ばれる。有名なペネトレーションテストツールにMetasploitがある[4]。

類似技術に脆弱性検査(Vulnerability Scanning)がある。これはNessus[7]などの脆弱性スキャナを利用して、脆弱性を含むバージョンのソフトウェアが使用されているかを確認する。脆弱性検査はツールなどで自動化が容易であるが、脆弱性を悪用してどのような攻撃が実現可能かまでは脆弱性検査で調べることができない。

制御機器に対するペンテストは、一部のセキュリティ会社において、制御システムセキュリティ診断試験などの名称でサービス提供されている[6]。しかしペンテストの品質は、ペンテスターのノウハウに強く依存するため、属人性が高く、ペンテストすべてを自動化することは難しい。

2.1 セキュリティ検査とコードカバレッジ

ペンテストをはじめとするセキュリティ検査での脆弱性発見件数は、検査中に実行したコードカバレッジに比例する事が経験則的に知られている[3]。そのため、セキュリティ検査においては、検査時に実行されるコードカバレッジが高くなるように検査を行うことが重要である。例えば、ソフトウェアの脆弱性を発見する、バイナリファザーであるAFL[9]は、コードカバレッジを評価関数とした遺伝的アルゴリズムをファズエンジンに用いることで、有名OSSの脆弱性を多数発見するような高い性能を実現している。

ではペンテストにおけるコードカバレッジはどのように考えればよいだろうか。システムが有する状態をどれだけ、状態遷移し実行したかをコードカバレッジと等価に考えることができる。この場合、検査対象システムが持ちうる状態をより多く遷移するような入力データを用いてペンテストを行う必要がある。また対象システムの状態遷移を特定することで、手動でペンテストする際の手がかりとして用いることが可能で、ペンテスト作業の効率化が期待できる。

つまり検査対象システムの状態遷移をより多く推定(特定)することで、ペンテストの効率化と品質の向上、さらには高い専門性を有するペンテスターへの依存を一部脱却することが期待できる。

3. 提案手法

本稿の提案手法について解説する。今回の提案手法の目的と前提は以下の通りである。

表 1　提案手法の目的と前提

目的	・対象システムに搭載されたソフトウェアの状態遷移を自動で推定する ・推定した状態遷移情報を用いてペンテストを効率化する
前提	・対象システムの内部(ソフトウェア)仕様はブラックボックス ・デバッガ等は接続不可 ・入出力インターフェースを有する(アナログ、デジタルは問わない)

今回おいた前提は一般的な制御システムや車載機器に相当する。機器が備える入出力インターフェースは、アナログ信号でもネットワークパケットなどのデジタル信号でもよい。例えばネットワークインターフェースから受け取ったパケットの特定のバイト値に応じた、アナログ値を電圧レベルとして出力するシステムが、今回前提とする典型的な制御システムとして考えられる。

3.1 状態遷移推定とシステム同定のアナロジー

入出力の相関から、システム内部の仕組みやモデルを推定する類似技術に、システム同定(System Identification)がある。システム同定とは、ブラックボックスまたはグレーボックス(ある程度の検討はつくが、詳細なパラメータなどが不明なシステム)なシステムに対して、システムを表す数理モデルを推定することである[8]。一般に対象システム

を表す数理モデルが未知、つまりブラックボックスモデルの場合には、適当に与えた入力信号に対する出力信号を記録し、入出力信号の時系列データから相関値算出や統計解析を行うことで、対象システムを表す数理モデルを推定する。有名な手法に ARX モデルや ARMAX モデルなどの多項式モデルを仮定して、対象システムに最もそれらしいパラメータとしてフィッティングする手法がある。

実際のシステム同定の場合には、入力値としてステップ関数や白色雑音など、基準となるテスト信号を入力した際の出力値を用いて解析を行う。例えば電気回路のシステム同定の場合には、入力のテスト信号を電圧として与え、出力として電圧もしくは電流を計測する。

3.2 状態遷移推定への適用

システム同定の考え方をペネトレーションテストに適用する場合を考える。システム同定では適当な入力値に対する出力値を観測し、入出力の相関や統計解析結果からシステムを表す数理モデルを推定していた。今回推定すべきものは、対象となる制御機器の状態遷移であり、システム同定の考え方と同様に入出力の相関からシステムの状態遷移を推定する。提案方式では、入出力の相関変化を状態遷移としてとらえ、状態遷移の推定を行う図 1 にシステム同定と、提案方式の類似性について示す。

次に提案方式の詳細について説明する。提案方式は 1. 入出力ポート相関推定フェーズ と 2. 状態遷移推定フェーズの 2 つのフェーズで構成される。

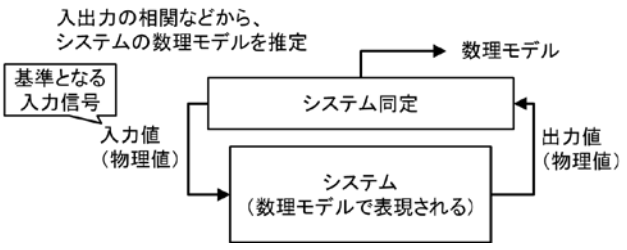
3.2.1 入出力ポート相関推定フェーズ

制御機器には入出力インターフェースが複数ずつ存在する。どの入力がどの出力に相関するか、入出力ポート自体の相関をはじめに推定する必要がある。例えば入力が A,B、出力が C,D と 2 入力 2 出力なシステムが対象だったとした場合、A に対する入力が、C,D どちらの出力に影響しているのか、または C,D 療法に影響するのかを特定する必要があり、入出力ポートの相関の推定はこれを意味する。

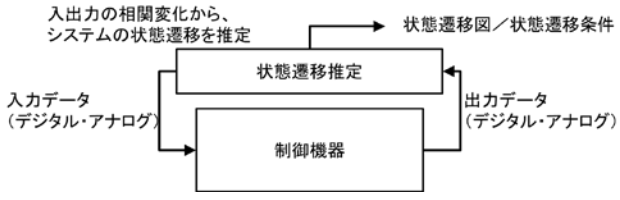
3.2.2 状態遷移推定フェーズ

入出力ポートの相関を推定した結果を用いて、状態遷移推定を行う。相関のある入出力ポートの組を用いて、状態遷移推定を実施する。この時の入力データはランダムに生成しても良いし、探索範囲が小さいならば全探索を行っても良い。入力データに対して得られた出力データは、入力データと一緒に時系列データとして記録する。次に得られた入出力データを用いての相関値を算出する。相関値の算出は相関係数などの形で算出することができる。相関値も時系列データとして記録し、時系列変化が一定以上であった時に、状態が遷移したと判断して、状態遷移を記録する。また状態遷移が発生した直前の入力を遷移条件として記録する。これらを繰り返すことで、状態遷移を推定する。

例を用いて説明する。ここで例示するシステムは、入力は 2 バイトのデジタルデータ、出力はアナログ値と仮定する。出力値は入力したデータの上位 1 バイトの値が制御モード、下位 1 バイトの値が制御値であり、制御モード(0x00 そのまま、0x0 ビット反転)に応じた制御値が、アナログ値として出力される仕様だと仮定する。今回の入力データ生成はインクリメンタルサーチで入力空間を探索する。この時の入出力の関係は、例えば図 2 のような入出力の時系列データが得られる。またこのときの入出力相関値を算出すると、図 3 に示すような相関値の時系列データが得られる。



(a) システム同定の概念図



(b) 提案方式の概念図

図 1　システム同定と提案方式とのアナロジー

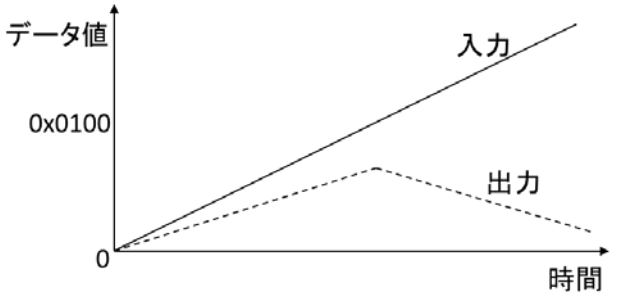


図 2　例示システムの入出力時系列データ

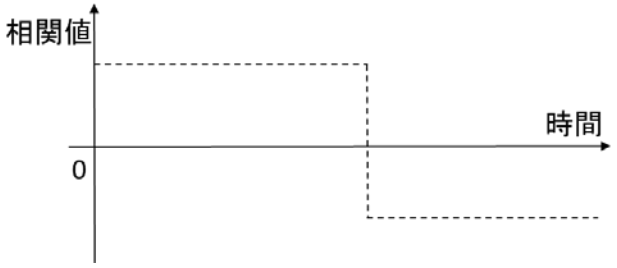


図 3　例示システムの時系列相関値変化

今回の例ではデータ値の上位 1 バイトが制御モードで、0x00 のときはそのままのバイト値、0x01 のときはビット反転した値を出力するシステムと仮定しているため、入力データが 0x00FF と 0x0010 を境に、システムの状態が変化する。入出力相関も同様に、同じタイミングで相関値に変化が発生する。この相関値の時系列変化から、システムの状態が変化したととらえて、状態遷移として記録する。この推定を繰り返すことで、システムの状態遷移を推定する。

4. 考察

4.1 提案方式の妥当性

今回の提案方式では、対象システムに入力する入力データに対して、出力データが何らかの相関があることと、状態変化が発生すると相関が変化するという、2 つの前提を置いた。果たしてこの前提は現実的なのだろうか。

実際の制御機器においては、制御値のやり取りを行う通信や入力以外に、ファームウェアや制御ラダーなどを書き換える機能がある。一般にその最中には、制御動作を停止し、プログラム書き換えを行うため、どのような入力を与えたとしても出力には反映されない。よって、状態変化を相関値の変化として捉えることができる。

4.2 入力データについて

システム同定では、モデルを推定するときに入力した信号は、ステップ信号などの基準となる信号を入力値として用いていた。今回の状態遷移推定における基準となる信号があるのだろうか。仮に状態遷移推定やセキュリティ検査を行う場合の基準信号があるとするならば、ファジングにおいて適当な入力データを生成する必要性が無くなってしまう。これは今回提案している状態遷移推定においても同様であり、基準となる入力データは存在しないと考えられる。その理由としては、システム同定で扱っているシステムそのものは、物理法則に従ったシステムであり、システムの数理モデル自体は、パラメータの微細な変化はあるものの、基本的には不変である。一方で、ペンテストで扱うシステムは、ソフトウェアなどで記述されたシステムであるため、その挙動は実装に依存すると言わざるを得ない。そのため、基準となる信号のみで、状態遷移推定を行う事は困難である。

効率的に状態遷移を多く行わせるためには、今回の提案方式の場合、相関変化が大きくなるような入力データ列を生成する必要がある。例えば相関値変化の大きさと評価関数と定義して、遺伝的アルゴリズムを適用すると、状態遷移を多く行わせる入力データ列の生成が期待できる。

4.3 状態遷移の重複記録について

システムの状態遷移は、遷移前の状態に戻ることも考えられる。例えば図 4 に示すような状態遷移図を考えると、S1 から S2、S3 と状態変化した後、再び S1 に状態遷移している。提案方式で状態遷移推定を行うと、状態遷移が起こるたびに新たな状態として記録され、同じ状態に戻ったにも関わらず、あたかも別の状態に遷移したように記録されてしまう。これを防ぐ方法として、遷移後の相関値も記録しておき、過去に記録した状態の相関値と類似する相関値に遷移した場合には、過去に記録した状態と同じ状態に遷移したものとして、重複記録しない方法が考えられる。

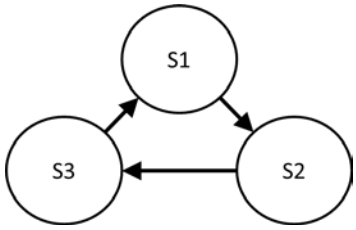


図 4　同じ状態に繰返し遷移する例

5. まとめ

本稿では制御機器のペネトレーションテスト支援のための、システム状態遷移推定方式を提案した。制御システムは、入力と出力が相関していることがほとんどであり、相関値が変化するという事は、システムの状態、つまり制御機器の制御モードなどの状態が切り替わったことを意味することが多い。そのため、入出力相関値の変化から状態遷移を推定できる。提案方式により、効率的なペンテストの支援が期待できる。

今後の課題として、実際の制御機器において相関変化を検知し状態遷移が正しく推定できることを評価する。また少ない入力データから多くの状態遷移を推定するための、効率的な入力データ生成方式についても検討を行う。

参考文献

[1] Kaspersky, Myrtus and Guava, Episode 1. https://securelist.com/myrtus-and-guava-episode-1/29614/ 2010-07-15(2018-05-07 閲覧)
[2] Charlie Miller, Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. 2015, 91p.
[3] Charlie Miller. Fuzz By Number. CanSecWest2008. https://cansecwest.com/scw08/scw08-miller.pdf 2008-03-28(2018-05-07 閲覧)
[4] David Kennedy et al, 青木一史ほか監訳. 実践 Metasploit. 2012, 368p.
[5] 情報処理推進機構. 制御システムのセキュリティリスク分析ガイド. 2017, 250p. https://www.ipa.go.jp/files/000061925.pdf (2018-05-07 閲覧)
[6] 株式会社サイバーディフェンス研究所. 制御システムセキュリティ診断試験. https://www.cyberdefense.jp/services/assessment_service/ics_testing.html. (2018-05-07 閲覧)
[7] Tenable. Nessus Professional. https://www.tenable.com/products/nessus/nessus-professional/ (2018-05-07 閲覧)
[8] 足立修一. システム同定の基礎. 東京電機大学出版局. 2009, 256p.
[9] American Fuzzy Lop. http://lcamtuf.coredump.cx/afl/ (2018-05-07 閲覧)