Vol.2013-DPS-154 No.31
Vol.2013-CSEC-60 No.31
2013/3/14

# パッチ情報を用いた IDS シグニチャ検証手法の検討

河内清人† 桜井鐘治†

IDS を用いてネットワーク監視を行っている組織にとって，IDS が使用しているシグニチャが，検知対象としている攻撃を漏れなく検知可能かどうかは重大な関心事項である．しかし，そのような情報は IDS ベンダによって提供されることは通常無いため，IDS のユーザが独自にシグニチャを検証する必要があった．従来シグニチャの検証では，実際に攻撃ツールを入手して攻撃トラフィックを生成して IDS に入力し，検知が行えるかどうかを判定する方法が知られているが，使用した攻撃ツールの生成するトラフィックが全ての攻撃パターンを網羅していない限り，シグニチャに検知漏れを起こす可能性が残ってしまう．
そこで，本稿では攻撃ツールに依存せずに，シグニチャに検知漏れの可能性があるかどうかを判定する方式を提案する．本方式では，脆弱性を持ったプログラムに対して公開されたパッチ情報から攻撃データのパターンを抽出し，シグニチャ上で定義されたパターンと比較する事で，シグニチャが攻撃データパターンを網羅しているか判定する．
本方式は現在検討段階である．今後，パッチ情報からの攻撃データパターン抽出誤りが発生した場合等への対応を検討し，試作・評価を実施する予定である．

# IDS Signature Verification Using Patch Information

KIYOTO KAWAUCHI† SHOJI SAKURAI†

Organizations which use IDS concerns themselves to confirm the signatures of the IDS can detect all the variant of the attacks which they are written for. However, the vendor of the IDS rarely provides the information about detectability of the signatures, and so, the users of the IDS must verify the signatures on their own efforts. It is known as the way of verifying signatures to test the IDS using exploit codes, but it is difficult to confirm the signature can detect all the variant of the attacks unless the exploit codes generates all pattern of traffic.
We propose the technique to verify signatures without exploit codes. It extracts the pattern of attack data from the published patch information for the vulnerable program, and compares the attack pattern and the pattern defined on the signature. It decides the signature covers all the pattern of the attack or not.
Currently, our proposal is under design phase. We will solve the remaining challenges, such as dealing with incorrectly extracted attack data pattern, and we will make a prototype and evaluate our proposal.

## 1. はじめに

侵入検知装置（IDS）では，シグニチャと呼ばれるパターン情報と監視対象ネットワークを流れるトラフィックとを比較し，パターンに合致する通信がネットワークを流れた場合に攻撃とみなす．シグニチャは，検知対象とする攻撃ごとに定義されており，シグニチャに合致する通信が発生した場合に，そのシグニチャが検知対象としている攻撃が発生したとして，ネットワーク管理者に通報が行われる．

シグニチャは，新たな攻撃が発見される度に IDS の開発元が作成し，定期的，あるいは必要に応じて装置のユーザに配布されるのが一般的である．

IDS を用いてネットワーク監視を行っている組織にとって，IDS が使用しているシグニチャが，検知対象としている攻撃を漏れなく検知可能かどうかは重大な関心事項であるが，そのような情報が，IDS ベンダによって提供されることは通常無く，IDS のユーザが独自にシグニチャを検証する必要があった．

ユーザがシグニチャを検証する方式としては，実際に攻撃ツールを入手して攻撃トラフィックを生成して IDS に入力し，検知が行えるかどうかを判定する方法が知られている[1]．しかし，使用した攻撃ツールの生成するトラフィックが全ての攻撃パターンを網羅していない限り，実際にはシグニチャに検知漏れの可能性があってもユーザは認識することができない．

そこで，本稿では攻撃ツールに依存せずに，シグニチャに検知漏れの可能性があるかどうかを判定する方式を提案する．本方式では，脆弱性を持ったプログラムに対して公開されたパッチ情報から攻撃データのパターンを抽出し，シグニチャ上で定義されたパターンと比較する事で，シグニチャが攻撃データパターンを網羅しているか検証する．

本方式では，攻撃データパターンは文脈自由文法として表し，シグニチャで定義されたパターンから，シグニチャでは検出されないデータパターンを表す有限状態オートマトン（以下オートマトン）を生成する．両者に合致するデータ列が存在するかどうかを検査する事で，検証を行う．

† 三菱電機株式会社 情報技術総合研究所
Mitsubishi Electric Corporation, Information Technology R&D Center

Vol.2013-DPS-154 No.31
Vol.2013-CSEC-60 No.31
2013/3/14

## 2. 関連研究

### 2.1 IDS シグニチャの検査

IDS シグニチャを検査する方式として Vigna らは，脆弱性を攻撃するサンプルデータ（Exploit テンプレート）を様々な形式に変形（mutate）し，評価対象の IDS に入力して検知結果を確認する方式を提案している[1]. 変形は, IDS の検知を回避するための攻撃データ変形テクニック（IDS evasion technique）として知られている幾つかの方式をテンプレートに適用することで行われる．このようなテクニックとしては，例えば IP パケットの分割，標的の Web サーバでは許容されるような軽微なプロトコル違反の HTTP メッセージへの注入などが挙げられている．

この方式では，テンプレートとして示された攻撃データを変形したデータを試験に使用しているが，最終的に脆弱性を含んだコードが処理するデータはテンプレートに示されたデータと同一となる．そのため，テンプレートとしたデータとは異なる攻撃データに対して IDS のシグニチャが検知可能であるかどうかは検証できない．

### 2.2 攻撃データ生成

脆弱性への攻撃データを自動生成する手法として[2]が提案されている．本方式では，パッチによって修正される前後のプログラムを静的に解析し，パッチによって新たに追加された条件分岐を抽出し，その後同分岐によって旧コード上では存在しなかった処理に分岐するような入力を生成する．

入力の生成は，プログラム中にある，外部からデータを受信する個所から追加された条件分岐までの制御フローに現れる条件分岐を取り出し，全ての条件を満たす入力を Constraint Solver を用いて求める．

このような技術を用いて攻撃データを生成し，実際に IDS へ入力する事で，IDS が使用するシグニチャの検証を行う事ができるが，攻撃データのバリエーションが膨大である場合，検証に膨大な時間がかかる恐れがある．

### 2.3 シグニチャ自動生成

Newsome らは，ポリモルフィックワームを検出するための IDS シグニチャ生成手法として, Polygraph[3]を提案している．Polygraph では攻撃データのサンプルを複数集め，それらに共通して含まれるデータ列を抽出してシグニチャとする．

一方，Brumley らは，[4]において，脆弱性を含むプログラム，脆弱性を発生するプログラム上のコード位置，脆弱性の発生条件およびサンプルの攻撃データを与える事で，攻撃データを入力されたプログラムの挙動を調査し，シグニチャを作成する方式を提案しているが，本稿の目的である既存シグニチャの検証には直接は適用できない．

## 3. 提案方式

### 3.1 システム概要

本提案方式のシステムの概要を図 1 に示す. 本システムは，シグニチャ抽出部，攻撃データパターン抽出部，検知対象外データパターン抽出部，パターン比較部で構成されている．

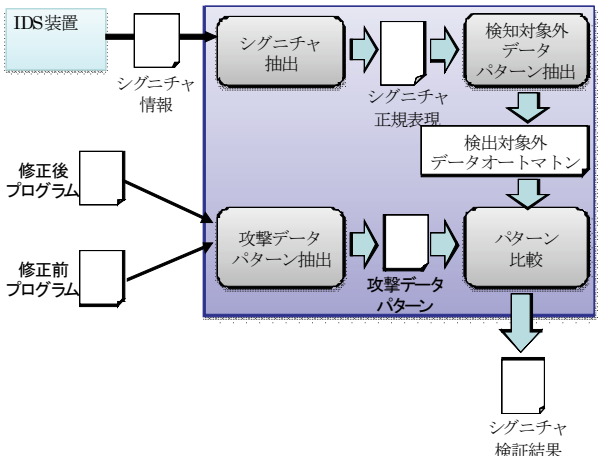


図 1　システム概略図

シグニチャ抽出部はシグニチャ情報から同シグニチャが検知対象とするデータのパターン(正規表現)を取り出す処理である．

攻撃データパターン抽出部は，修正前プログラム及び修正後プログラムを入力とし，脆弱性に対処するためにプログラムに施された修正内容から，脆弱性を攻撃可能な攻撃データのパターンを生成する．

検知対象外データパターン抽出部は，シグニチャ抽出部によって取り出された正規表現から，同正規表現にマッチしない全てのデータ（検知対象外データ）にマッチするオートマトンを生成する．

最後に，パターン比較部は，検知対象外データパターン抽出部が出力したオートマトンで受理されるデータの集合と，攻撃データパターン抽出部が出力した攻撃データパターンを満たすデータの集合の積をとり，結果が空集合となるかどうかを判定する．

結果が空集合だった場合，攻撃データはすべてシグニチャによって網羅されているとし，そうでない場合には，一部の攻撃データがシグニチャの検知対象から漏れていると判定し，比較結果に基づいてシグニチャで検知できない攻撃データ例を出力する．

以降の節ではこれらの機能のうち，以下の機能についてより詳しく説明する．

- 攻撃データパターン抽出
- 検知対象外データパターン抽出
- パターン比較

Vol.2013-DPS-154 No.31
Vol.2013-CSEC-60 No.31
2013/3/14

### 3.2 攻撃データパターンの抽出

**攻撃データパターン抽出**

入力:
P←修正前プログラム
P'←修正後プログラム

出力:
攻撃データパターンを示す文脈自由文法

手続:
1: C ←攻撃データの制約条件算出(P, P')
2: C' ←加法標準形に変換( C )
3: G ← ( {*S*}, Σ, ϕ , *S* )
4: for n = 1 to C'の節数
5: N ← new_nonterminal()
6: X ← ({N}, Σ ,{ N ::= ε , N ::= <任意の一文字> N },N)
7: for m = 1 to C'の n 番目の節のリテラル数
8: R ← make_automaton( C'[n][m] )
9: X ← intersect( X, R )
10: end loop
11: G.生成規則←G.生成規則 ∪ { X.生成規則, *S* ::= N }
12: G.非終端記号 ← G.非終端記号 ∪ N
13: end loop
14: return G

図 2 攻撃データパターン抽出疑似コード

前述の通り，攻撃データパターン抽出部は，修正前プログラム及び修正後プログラムから，脆弱性に対処するためにプログラムに施された修正個所を特定し，その内容から脆弱性を攻撃可能な攻撃データのパターンを生成する．

図 2 は，攻撃データパターン抽出部の動作例を示す疑似コードである．

疑似コードに示すとおり，攻撃データパターン抽出部は，修正前プログラム P 及び修正後プログラム P’を入力として呼び出される．

まず，ステップ 1 において，攻撃データパターン抽出部は，脆弱性を攻撃するデータに関する制約条件を算出し，変数 C へ代入する．制約条件の算出は，[2]で示しているように，P，P’を比較して，脆弱性への対策として追加された入力データ検査処理を抽出し，データの入力から同検査処理に到達するまでに行われる（既存の）入力データ検査の検査条件と，新たに追加された入力データ検査の検査条件を組み合わせることで算出する．

ステップ 1 で生成される制約条件は，例えば以下のようなものになる．

(入力文字列長 < 100)∨ (100 ≤
入力文字列長 ∧ 入力文字列内に"mode=LONG" を含む)

次に，ステップ 2 において，生成された制約条件 C を加法標準形に変形し，変数 C’に格納する．加法標準形とは論理式を連言節の選言として表現したものであり，任意の論理式をこの形式に変形することが可能であることが知られている[5].

次に，ステップ 4 からステップ 14 で，C’で示す制約条件を満たす文字列の「文法」を生成する．文法は文脈自由文法として表される．文脈自由文法とは，全ての生成規則が，左辺が 1 個の非終端記号のみで構成されている文法である[6]．文脈自由文法は，一般に次の 4-tuple であらわされる．

G = ( N, Σ, R, S )
- N …… 非終端記号の集合
- Σ …… 終端記号の集合
- R …… 生成規則の集合
- S …… 開始記号

まずステップ 3 で開始記号 *S* のみを含み，生成規則を一つも含まない文法で G を初期化する．

次に, ステップ 4 から 13 のループにおいて, 制約条件 C’中の各連言節に対応する制約条件を満たした生成規則を生成する．

まず，ステップ 5～6 において，変数 X を，任意の文字列を受理する文法に初期化する．ステップ 5 では，新規の非終端記号を生成して変数 N に代入し，同非終端記号を開始記号とし，同記号が，任意の文字列を受理するような生成規則の集合({N ::= ϵ, N ::= < 任意の 1 文字 > *N*})で規定される文法を，ステップ 6 で変数 X に代入している．

次に，ステップ 7～10 に示すループにおいて，X を n 番目の連言節に含まれる全ての制約条件を満たす文法に変形する．

まず，ステップ 8 において， C’の n 番目の連言節に含まれる m 番目のリテラル（原子論理式及びその否定）を取り出し，取り出された条件を対応するオートマトンに変換し，変数 R へ代入する．

リテラルからオートマトンへの変換は変換テーブルを用いて行う. 変換テーブルは, 変換対象とする条件に対し，対応するオートマトンを生成するための方法を定義したテーブルである．変換テーブルは変換対象とする条件を記述したカラムと対応するオートマトンの生成方法を定義したカラムで構成されている．

変換テーブルの例を図 3 に示す．なお，図中に示されているオートマトンでは，二重丸で表された状態が受理状態を示している．

例えば“入力文字列長<100” というリテラルに対しては，条件 1 に合致するため図中(1)に示されるようなオートマトンが生成される. ただし, 図中 N と表記されている部分は，この例の場合 100 になる．

Vol.2013-DPS-154 No.31
Vol.2013-CSEC-60 No.31
2013/3/14

| | 条件 | 対応するオートマトン |
|---|---|---|
| 1 | 入力文字列長 < N | (1)に示すオートマトンを生成する |
| 2 | 入力文字列内にSTRを含む | (2)に示すオートマトンを生成する |
| 3 | 「条件」の否定 | 「条件」に対応するオートマトンの受理可・不可を反転 |
| … | … | … |

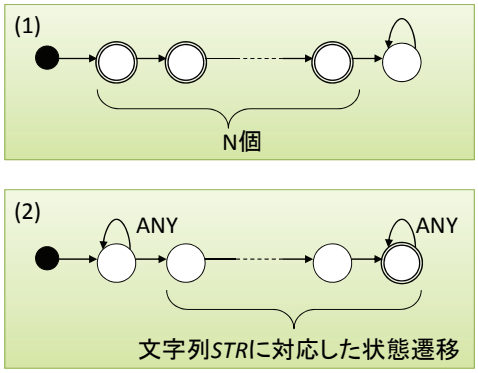


図 3　オートマトンへの変換テーブル

リテラルがある原子論理式の否定であった場合には，同原子論理式に対応するオートマトンを生成し，その後同オートマトンの受理・非受理状態を反転させる事で目的のオートマトンを得る．

次に，ステップ 9 において文法 X とオートマトン R を引数として手続き intersect が呼び出される．

intersect は，文法 X とオートマトン R がともに受理する文字列が満たす文法を生成する処理である．そのアルゴリズムとしては，Reps らが示した方式[7]が知られている．

intersect の処理の概要は以下の通りである．まず，intersect の対象とする文脈自由文法を正規化する． 正規化は，各生成規則の右辺がたかだか 2 記号となるように，適当に新たな非終端記号を追加しながら元の生成規則を変形する処理である．

次に，正規化された文脈自由文法の各生成規則に対応するオートマトン上のパスに，同生成規則の左辺となる非終端記号をラベル付けしていく．最終的に，オートマトンの開始状態から受理状態の間に，開始記号が対応付けられたら終了となる．

オートマトンと文脈自由文法の intersect 処理の例を図 4 に示す．出力された文脈自由文法の各非終端記号の添え字は，オートマトン上のどのノードからどのノードへの遷移に対応するものかを示している．

ループを終了した時点で，文法 X には，C’ 中の n 番目の連言節内の全てのリテラルとの intersect をとった文法が格納されているので，ステップ 11,12 でこの文法を文法 G に登録する．登録は文法 X 内の全ての非終端記号を，G 内の非終端記号集合に追加し，さらに X 内の全ての生成規則を文法 G に追加する事で行う．その後，G の開始記号 *S* の生成規則として *S*::=N を追加することで行われる．これにより，文法 G は，n 番目の連言節を満たす文字列が受理さ

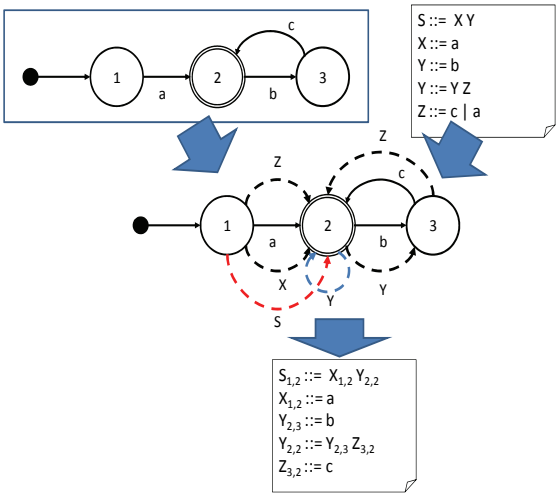


図 4　intersect 処理の例

れるように修正される．その後，再度外側のループに戻り，次の連言節に対して同様の処理を行う．

以上の処理を行うことで，最終的に文法 G は，制約 C’ で示された制約条件を満たし得る全ての文字列を表す文法が格納され，ステップ 14 にて同文法を処理結果として出力する．

### 3.3　検知対象外データパターン抽出

前述の通り，シグニチャ検知対象外データパターン抽出部は，シグニチャに記載された攻撃データを表す正規表現から，同正規表現にマッチしない全てのデータ（検知対象外データ）を表すオートマトンを生成する．検知対象外データパターン抽出処理の疑似コードを図 5 に示す．

```
検知対象外データパターン抽出
入力:
S ← シグニチャから抽出された正規表現

出力:
検知対象外データパターンを示すオートマトン

手続:
1:  S’ ← “.*” || S || “.*”
2:  A ← regex_to_automaton( S’ )
3:  A’ ← A の受理状態と非受理状態を反転
4:  return A’
```

図 5 検知対象外データパターン抽出疑似コード

本処理ではまず，入力としてシグニチャ情報のパターン定義情報を表す正規表現 S を受け取る．

次にステップ 1 において，S の両端に任意の文字列を受理する正規表現である”.*” を付加した新たな正規表現 S’ を生成する．これは，通常 IDS では，シグニチャで定義さ

れる正規表現にマッチするデータが，通信データのどこに存在しても攻撃とみなすことに対応している．

その後，ステップ 2 において，正規表現 S’から，同正規表現にマッチするデータのみを受理可能なオートマトンを生成し，変数 A に格納する．

与えられた正規表現にマッチするデータのみを受理可能な有限状態オートマトンの構成方法は広く知られており，例えば[6]などに詳しい．

オートマトン生成後，ステップ 3 において，変数 A に格納されたオートマトンの受理状態と，非受理状態を入れ替えることで新たなオートマトンを生成（A’）し，それを出力して処理は終了する．

入力された正規表現に対し，どのように処理が行われるかの具体例を図 6 に示す．この例ではシグニチャ情報の正規表現として”abc”を用いている．最終的に”abc”を含まない全ての文字列を受理するオートマトンが生成されていることが確認できる．

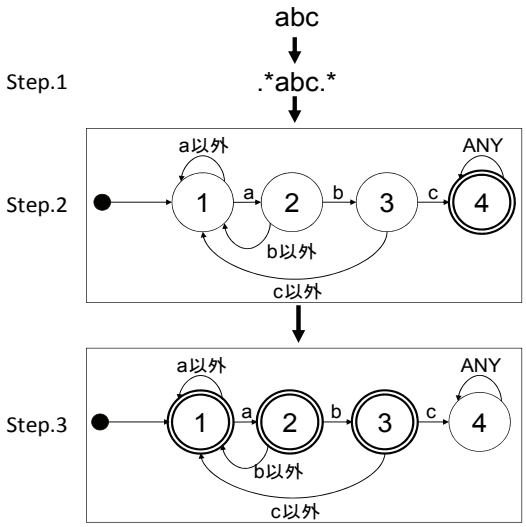


図 6 検知対象外データパターン抽出例

### 3.4 パターン比較

前述の通り，パターン比較部は，シグニチャ検知対象外データパターン抽出部によって算出された検知対象外データのパターン（オートマトン）と，攻撃データパターン抽出部によって算出された攻撃データパターン（文脈自由文法）を比較することで，シグニチャでは検知できない攻撃データの有無を判定する．

パターンの比較は攻撃データパターンと検知対象外データパターンを intersect に入力する事で行う．intersect が失敗した場合，両パターンを満たすデータパターンは存在しないということであり，従って全ての攻撃データパターンはシグニチャで検知されると判定して良い．一方，intersect が成功した場合，シグニチャで検知されない攻撃データのパターンを示す文法が存在することとなるため，シグニチャに漏れがあると判定する．

## 4. 考察

### 4.1 攻撃データパターン抽出誤りへの対応

今回の提案方式では，攻撃データパターンの抽出に[2]で提案されている，修正パッチによって追加された条件分岐ロジックに基づく方式を利用している．そのため脆弱性とは無関係な修正があわせて行われた場合に，脆弱性を攻撃する事とは無関係なデータが攻撃データとして抽出されてしまい，その結果，実際には問題の無いシグニチャに対して検出漏れがあると誤判定してしまう可能性がある．

このような判定誤りを防ぐために，パターン比較によって抽出された攻撃データの文法に基づいて攻撃データを生成し，実際に脆弱性が攻撃されるかを検証することが考えられる．具体的には得られた文法に対して[8]に示すような文生成アルゴリズムを適用することで実現が可能になると考えられる．

その場合，入力した攻撃データにより脆弱性が再現したかどうかの判定は，オペレータによる目視確認が必要になると考えられる．

### 4.2 オートマトン変換テーブルの表現力向上

今回の方式では，攻撃データパターン抽出において，抽出された入力データへの制約をオートマトンとして表現する．これは，intersect で示されるアルゴリズムから要請される制限である．オートマトンで表現可能な言語は正規表現で表現可能な言語と等価であるため，制約によってはオートマトンで表現できない場合も考えられる（例えば文脈自由文法でのみ表現可能な文字列パターン等）．

このような場合に対応するため，入力データへの制約は文脈自由文法として生成するようにし，make_automaton の実行終了時にオートマトンへ近似変換する方法が考えられる．変換には，[9]で示されているようなアルゴリズムが提案されている．しかし，このような変換により，元の制約には含まれないパターンのデータも含まれるようになる（制約が緩やかになる）ため，最終的に算出される攻撃データパターンにも，実際には無害なデータが含まれるようになる．

### 4.3 シグニチャの適用範囲に対する制約への対応

IDS 製品によっては，シグニチャは単純な正規表現だけでなく，シグニチャの適用範囲を指定可能である．例えば snort[10]においては，ネットワーク上を流れる通信データ中のどの範囲に対しシグニチャを適用すべきかをオフセットとして指定する事ができる．

現在の方式においては，上記の例にあるような，正規表

現として与えられたデータパターン以外に加えられた制約は考慮していない．そのため，追加された制約が不適切であることによる検知漏れの有無については判定することができない．

改善手法としては，各 IDS 製品のシグニチャごとに，追加で与えられている制約条件を読み取り，それを元々の正規表現に付加する必要があると考えられる．

## 5. まとめ

IDS のシグニチャが，検知対象としている攻撃を全て検出できるかどうか，IDS による監視を実施する組織において，シグニチャを検証する場合，実際に攻撃ツールを入手して攻撃トラフィックを生成して IDS に入力し，検知が行えるかどうかを判定するしかなく，使用した攻撃ツールの生成するトラフィックが全ての攻撃パターンを網羅していない限り，実際にはシグニチャに検知漏れの可能性があった．

そこで，本稿では脆弱性を持ったプログラムに対して公開されたパッチ情報から攻撃データのパターンを抽出し，シグニチャ上で定義されたパターンと比較する事で，シグニチャが攻撃データパターンを網羅しているか検証する方式を提案した．

本方式ではまず，Brumley らの方式を応用し，パッチによる修正前後のプログラムの差分から脆弱性を攻撃するデータの制約条件を抽出し，それらを満たす文脈自由文法を生成して，攻撃データパターンとする．並行して，シグニチャから取り出された正規表現をオートマトンへ変形し，さらに，同オートマトンからシグニチャでは検出されないデータを受理するオートマトン（検知対象外データパターン）を生成する．

最後に，生成された攻撃データパターンと，検知対象外データパターンとの intersect をとる．両データパターンに合致するデータが存在する場合には，シグニチャに漏れがあるとみなし，利用者へ通知する．

今後取り組むべき課題は，(1) 攻撃データパターンの抽出誤りへの対応，(2) オートマトン変換テーブルの表現力向上および (3) シグニチャの適用範囲に対する制約への対応が考えられる．

本方式は現在検討段階である．今後，試作を行い評価を実施していく予定である．

## 参考文献

1) G.Vigna,W.Robertson,D.Balzarotti:Testing Network-based Intrusion Detection Signatures Using Mutant Exploits, In Proc. of Computer and communications security (CCS '04). ACM,2004
2) Brumley, D.; Poosankam, P.; Song, D.; Jiang Zheng; , "Automatic Patch-Based Exploit Generation is Possible: Techniques and Implications," Security and Privacy(SP), 2008.. IEEE Symposium on , vol., no., pp.143-157, 18-22 May 2008
3) Newsome, J.; Karp, B.; Song, D.; , "Polygraph: automatically generating signatures for polymorphic worms," Security and Privacy, 2005 IEEE Symposium on , vol., no., pp. 226- 241, 8-11 May 2005
4) Brumley, D.; Newsome, J.; Song, D.; Hao Wang; Jha, S.; , "Theory and Techniques for Automatic Generation of Vulnerability-Based Signatures," Dependable and Secure Computing, IEEE Transactions on , vol.5, no.4, pp.224-241, Oct.-Dec. 2008
5) 家村 道雄 他 “入門電子回路（ディジタル編）” オーム社, ISBN 978-4-274-20365-7
6) 中田育男: コンパイラの構成と最適化, ISBN978-4-254-12139-1
7) D. Melski and T. Reps: Interconvertbility of set constraints and context-free language reachability. In Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation, pages 74–89,1997.
8) Lixiao Zheng; Duanyi Wu: A Sentence Generation Algorithm for Testing Grammars, Computer Software and Applications Conference, 2009. COMPSAC '09. 33rd Annual IEEE International , vol.1, no., pp.130-135, 20-24 July 2009
9) Mehryar Mohri and Mark-Jan Nederhof. Robustness in Language and Speech Technology, chapter 9: Regular Approximation of Context-Free Grammars through Transformation. Kluwer Academic Publishers, 2001.
10) Source Fire: Snort::Home Page, http://www.snort.org/, 2013 年 1 月 17 日確認