2009-IOT-4 (19)
2009 / 3 / 5

IDSに特化した文字列探索アルゴリズム

今泉 貴史[†]　　水野 恵祐[‡]

侵入遮断・検知システムにおいて、誤検知は完全にはなくすことができない。つまり、シグネチャマッチングにほんの少しの誤りが含まれていても、システム自身の検知率に関する性能の劣化はほとんど無い。ところが、この許容できる誤りを許すことで、シグネチャマッチングの速度に関する性能を大幅に引き上げられる可能性がある。

本論文では、誤りの発生を許容する代わりに文字列探索を高速に行う手法を提案する。厳密な探索ではないが処理が高速なアルゴリズムを多重化させて、誤りを運用上許容できる程度にまで減らす。この手法を用いることで、IDS のシグネチャマッチングを高速化できる。

String Matching Algorithms specialized for IDS

IMAIZUMI, Takashi[†]　　MIZUNO, Keisuke[‡]

While we use intrusion detection system, we can not be free from false positive alerts. In other words, there is little weakening concerning the detection rate of the system even if some mistakes are included at the signature matching. However, we allow some mistakes, and there is a possibility that the speed of the signature matching is greatly improved.

In this paper, we propose the technique for string matching at high speed instead of allowing the occurrence of the mistake. We multiplex non-strict high-speed algorithms to get allowable results. Using this technique, we can increase speeds of intrusion detection systems.

1　はじめに

インターネットが普及するにつれ、悪意を持ったユーザの存在は無視できないものとなっている。インターネットに接続された計算機は、常に不正侵入の危険にさらされているといっても過言ではない。これらの脅威から計算機を守るには、侵入検知システムや侵入遮断システムが有効である。侵入遮断システムでは、脅威となる通信を検知し、自動的にその通信を遮断することで、ネットワークを安全に守る。このシステムは、単にネットワークの入り口に設置すればよいため、利用者に特別な知識を要求しない運用も可能である。

現在多く利用されている侵入遮断システムや単に検知を行う侵入検知システムは、シグネチャマッチング方式により脅威となる通信を検知している。このシグネチャマッチングは、基本的には送信されるパケットの中にシグネチャと呼ばれるあらかじめ準備された文字列やパターンが含まれるかどうかを調べることである。しかし、この処理は計算機への負荷が高く、ネットワークの通信量が増えるに伴って、リアルタイムにすべての処理を行うことが困難になってきている。そのため、比較するシグネチャの量を減らしたり、すべてのパケットについて処理を行うのをあきらめたりすることも増えている。そのため、処理時間において占める割合が最も大きい文字列探索の高速化を図る手法が研究されている。

一方、シグネチャマッチング方式の侵入遮断・検知システムでは、誤検知を完全になくすことはできない[1]。これまで、誤検知を除去するために多くの研究がなされてきた。しかしそのいずれもが完全に満足のいく結果を出せてはいない。多くの研究では、侵入検知

[†]千葉大学 総合メディア基盤センター。Institute of Media and Information Technology, Chiba University.
[‡]奈良先端科学技術大学院大学 情報科学研究科。Graduate School of Information Science, Nara Institute of Science and Technology.

システムの発する警告を、学習手法などを用いながらさらに分類・処理することにより、ユーザが欲している警告かどうかを調べ、実際に警告するかどうかを判断している [2]。この手法では、ユーザが望む動作に対応できるように学習操作が必要になり、誰にでも簡単に使えるとはいえない。また、侵入操作を検知することに加え、さらに警告の選択処理を行う必要があり、ネットワークの広帯域化に伴う処理能力不足の問題を解決することはできない。

見方を変えると、侵入遮断・検知システムに関しては、シグネチャマッチングにほんの少しの誤りが含まれていても、システム自身の誤検知率に関する性能の劣化はほとんど無いことになる。この許容できる誤りを許すことで、シグネチャマッチングの速度に関する性能を大幅に引き上げることができる可能性がある。これが可能になれば、侵入遮断・検知システムにおいて、単位時間当たりに処理できるパケットのサイズや量を現状に比べて増やすことが可能となり、より高速・大容量のネットワークに対しても対応することが可能となる。

本研究では、誤りの発生を許容する代わりに文字列探索を高速に行う手法を提案する。厳密な探索ではないが処理が高速なアルゴリズムを多重化させて、誤りを運用上許容できる程度にまで減らす。

2 文字列探索アルゴリズム

文字列探索とは、長さ N のテキストと長さ M のパターンが与えられたときに、テキスト中にパターンが現れるかどうかを判定し、現れる場合にはその位置を示すことである。文字列探索アルゴリズムには、一度に単一のパターンを探索する single-pattern matching アルゴリズムと、一度に複数のパターンを探索する multi-pattern matching アルゴリズムがある。

single-pattern matching アルゴリズムとしては、Boyer-Moore 法 [3] や Knuth-Morris-Pratt 法 [4]、Rabin-Karp 法 [5] などが有名である。Knuth-Morris-Pratt 法では、各文字を 1 度だけ調べれば済むように事前に計算を行う。Boyer-Moore 法は、テキストに対してパターンの末尾から比較を行い、スキップによって可能な限り比較の回数を減らして高速化を図る。Rabin-Karp 法は、ハッシュ値を利用して探索を行うアルゴリズムである。このような single-pattern matching アルゴリズムを複数のルールについて繰返し適用することで、multi-pattern matching を実現することも可能である。しかし、この手法は、単に single-pattern matching を順次行っているに過ぎないため、探索に要する時間はパターン数に比例する。複数のパターンを探索する場合、multi-pattern matching アルゴリズムの方が効率的である。

multi-pattern matching アルゴリズムの多くは、与えられたパターンセットから事前にデータ構造を作成しておき、それを利用して全てのパターンについて同時に探索する。multi-pattern matching アルゴリズムとしては、Aho-Corasick 法 [6] や Commentz-Walter 法 [7]、Wu-Manber 法 [8] などが広く利用されている。これらのアルゴリズムは、探索のためのデータ構造としてトライ木や有限オートマトンを構築するのでメモリ使用量が多い。メモリ使用量の少ない multi-pattern matching アルゴリズムも提案されているが、その多くは処理速度が十分ではない。一般的に、multi-pattern matching アルゴリズムにおいて、処理時間とメモリ使用量はトレードオフの関係にある。

IDS に関連した、文字列探索を高速化するための手法としては、ExB 法 [9] のような pre-matching アルゴリズムも提案されている。pre-matching アルゴリズムは、単純な探索で「パターンが含まれる可能性」のみを検査する。pre-matching によって「パターンを含む可能性がある」と判定された場合のみ、Boyer-Moore 法などで完全な探索を行う。このように、pre-matching は完全な探索の回数を減らすことで、全体の処理時間の短縮を図る手法である。

pre-matching アルゴリズムを用いる場合、worst-case においてもアルゴリズム自体の処理時間が極端に増加することはないが、「パターンを含む可能性がある」とみなす可能性が高くなる。そのため、完全な探索の回数をほとんど減らすことができずに、pre-matching そのものが全く余計な処理となってしまう。pre-matching は average-case における処理の高速化を図る手法といえる。

さらに、ネットワークプロセッサや GPU、FPGA などを利用し、ハードウェアで文字列探索を行う研究も多い [10–13]。処理のパイプライン化や並列化、マルチスレッディングなどによって高速化を図る手法である。文字列探索アルゴリズムをハードウェア上で実装する

ためには、メモリ使用量の削減が最大の課題となっている。

3 IDSに特化した文字列探索アルゴリズム

3.1 IDSにおける文字列探索アルゴリズム

IDSにおいて、処理するパケットやシグネチャの数が増加すると多大な負荷がかかり、全てのパケットを処理しきれなくなってしまう。過負荷の状況では、IDSがパケットの取りこぼしを発生させるため、結局不正アクセスを見逃す可能性が高くなる。このようなIDSの *algorithmic performance vulnerability* を補強するためには、処理のボトルネックとなっている文字列探索を高速化する必要がある。IDSの検査処理には、性能に関して以下のような要件が求められる。

- average-caseにおいて高速に処理できる
- worst-caseでも極端に処理時間が増加しない
- 大規模なパターンセットに適用できる

しかし既存の文字列探索アルゴリズムは、テキストの各文字に対して厳密な探索を行うため，潜在的に高い *algorithmic performance vulnerability* を有している。探索結果の正確さを保証するために多大な計算資源を必要とし、IDSの性能向上を困難にしているのである。

しかしIDSの検知結果には通常誤検知が含まれているため、IDS自体の検査が厳密ではないといえる。そのため、処理のめざましい高速化が可能であれば、文字列探索の厳密さの低下は許容できると考えられる。

3.2 アルゴリズムフレームワーク

本研究では、厳密さの低下を許容して文字列探索を高速化するMI (Multiple Inspection) 法を提案する。この手法は、限定的な探索だが高速に処理できるアルゴリズムを多重化させて文字列探索を行う。個々のアルゴリズムでは探索しきれない範囲を多重化によって補完し、処理効率の向上を図るとともに、運用上許容できる程度にまで誤りの発生確率を抑える。

「厳密さの低下」を、文字列探索における「パターンでない文字列をパターンとみなす」誤りと定義する。

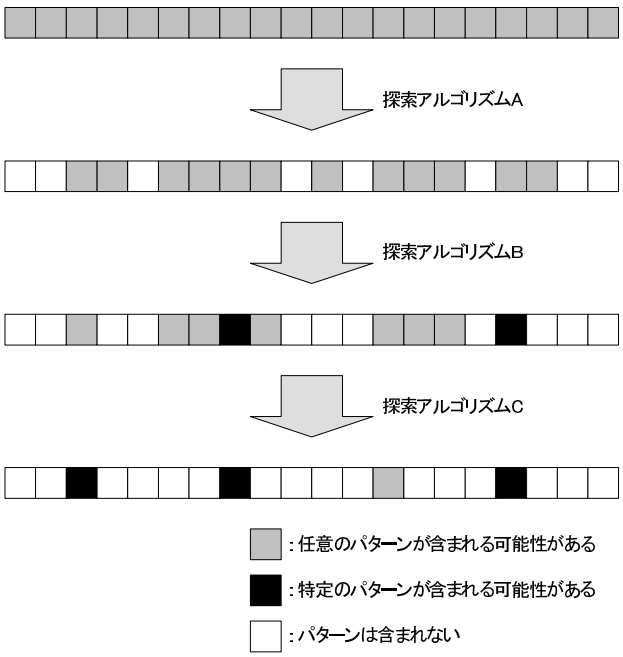

図 1: アルゴリズムの動作

IDSでの利用を考える場合、脅威でないパケットを脅威とみなしてしまうfalse positiveは許容可能な誤りである。一方、脅威のパケットを脅威として検知できないfalse negativeは、ネットワークの安全性を確保するために許すのは適当ではない。そのため本研究においては、「テキストに含まれるパターンを発見しない」誤りは許容しない。

フレームワークで利用する探索アルゴリズムは、テキストとパターンの集合が与えられたときに、テキストの各位置に対してパターンを含むかどうかを検査する。結果としては、テキストの各位置に対して

- 任意のパターンが含まれている可能性がある
- 特定のパターンが含まれている可能性がある
- パターンは含まれない

を返す。

さらに、探索アルゴリズムは他のアルゴリズムの出力を利用して動作するため、入力としても、出力と同じものを受け入れなければならない。最初に適用する探索アルゴリズムへの入力は、テキストの全ての位置に対して「任意のパターンが含まれている可能性がある」として扱う。最終的な出力は、最後のアルゴリズムの出力を用いる (図1)。

3.3 サンプルインプリメンテーション

アルゴリズムフレームワークの有効性を確認するために、サンプルインプリメンテーションを作成した。今回は、文字列探索アルゴリズムとして SOG 法 [14] を MI 法に適合させた SOG-like 法と、Rabin-Karp 法 [5] と似たハッシュを用いたアルゴリズムを利用した。この SOG-like 法とハッシュ値を用いた探索を MI 法に適用したものを、MI/SH と呼ぶ。

3.3.1 SOG-like 法

SOG(Shift-Or with q-Gram) 法 [14] は，Shift-Or 法を multi-pattern matching のために拡張したアルゴリズムである。q-Gram でアルファベットを増やしており、大規模なパターンセットへの適用も可能である。

SOG 法では、各パターンに含まれている全ての q-Gram について照合テーブル $B[c^q][M]$ を初期化すること以外は Shift-Or 法と同様に照合を行う。しかしこの方法では、B は複数のパターンの情報を保持しているため、パターンの出現と判定されても実際は該当するパターンが存在しない可能性がある。

SOG-like 法は、SOG 法にさらに修正を加えて「照合の候補」を絞り込めるようにしたアルゴリズムである。SOG 法は、扱う全てのパターンの長さを等しいと仮定している。しかし、IDS のシグネチャに記述されている各パターンの長さは広範囲に分布しているため、従来の SOG 法ではシグネチャのパターンを統一的に扱うことは困難である。

SOG-like 法は、SOG 法に対して以下に示す修正を加え、長さの違うパターンを統一的に扱えるようにした。

- 各パターンの先頭 8 文字を抽出して $M = 8$ のパターンセットを作成する。SOG-like 法は作成したパターンセットに対して SOG 法と同様の照合を行う。
- パターンセットには長さが 8 未満のパターンも含まれている。SOG-like 法では照合テーブル B を修正し、各 q-Gram についてパターンの末尾であるという情報を保持する。

また、長さが 8 未満の短いパターンからできるだけ多くの q-Gram を生成するために、overlapping q-Gram を利用する。

これらの修正により、SOG-like 法は「照合の候補」として「パターンを含む可能性のある位置」と「含まれる可能性のあるパターンの長さ」の情報を保持できるようになる。具体的には、テキストの各位置で行った検査の結果に対して、

1. 8 文字分の照合が発見された場合
   ⇒「この位置には長さが 8 以上のパターンが含まれる可能性がある」
2. 8 文字未満の照合中にパターンの末尾が出現した場合
   ⇒「この位置には長さが 8 未満のパターンが含まれる可能性がある」
3. パターンの末尾が出現せずに 8 文字分の照合も発見されなかった場合
   ⇒「この位置には明らかにパターンは含まれない」

という基準で「照合の候補」を絞り込む。

3.3.2 ハッシュ値を用いた探索

ハッシュ値を用いた探索アルゴリズムでは、SOG-like 法の照合結果から各候補のハッシュ値を求め、各パターンのハッシュ値と比較する。具体的には、パターンの長さおよびハッシュ値をキーとする探索処理を行う。

各パターンの特定にハッシュ値を利用するため、探索を 1 文字ずつ正確に行わなくてよい。また、各パターンの長さを保持する本来の目的は、SOG-like 法から渡される「含まれる可能性のあるパターンの長さ」を利用して探索処理の回数を減らすことである。しかし、パターンの長さもハッシュ値と同様に照合の十分条件として利用できるため、キー値に加えることによって衝突の可能性を低下させ、照合誤りの発生を減らしている。

3.4 実験

実装したプログラムの処理時間および照合誤りの発生について測定実験を行った。パターンの数および長さは、Snort のルールセットを参考にして、長さ 2 以上のパターンを 5,014 個作成した。照合対象のテキストは、長さ 1,460 で 10,000 個を準備した。なお、各パターンおよびテキストは、ASCII コード“ ”(0x20)～

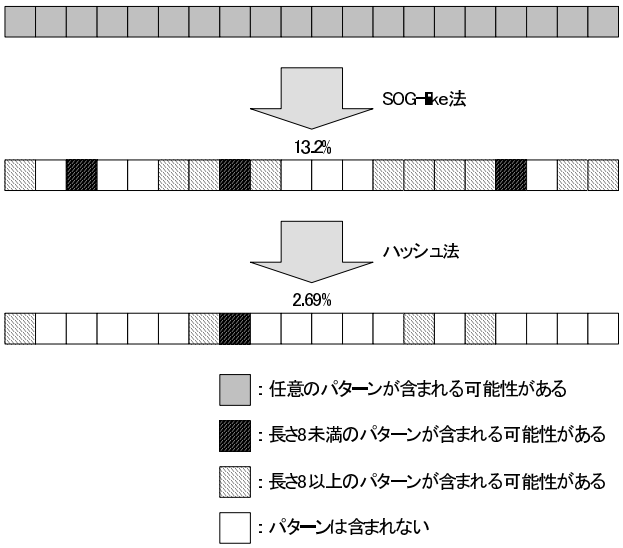

図 2: 実験結果

“~”(0x7e) の 95 種類の文字をランダムに並べた物を用いた。SOG-like 法の段階で、各テキストに対して平均 13.2%、さらにハッシュ値照合によって 2.69%にまで可能性の候補を絞り込んだ。今回のこの結果には、誤りは含まれなかった。

総当り法のプログラムを作成し、処理時間を比較した。総当たり法で探索を行ったところ、処理時間は 455.266 秒かかった。一方、提案手法では 7.781 秒であった。提案手法の場合、事前に照合用のデータ構造を作成する必要があるが、それに要した時間は 0.015 秒である。以上の結果より、十分に高速に動作することが確認できた。

4 考察

提案手法では、文字列探索の結果に誤りが発生することを許容している。しかし、誤りの発生確率は極めて低いことが本手法の前提である。ここでは、SOG-like 法とハッシュ値照合の 2 重化アルゴリズムを例にとり、提案手法が本実験において照合誤りを発生させる確率を概算する。計算を簡単にするために、提案手法を適用する環境について数値を簡単化するなどの仮定を行っている。

SOG-like 法で生じる誤りは、次の通りである。

- ビット配列 B の全エントリ数 $Entry_B$

$$Entry_B = 2^7 \times 2^7 \times 7$$

- パターンセットが使用するエントリ数 $Entry_P$

$$Entry_P = 8,192 \times 7$$

- 任意の 2-Gram が照合とみなされる確率 $Prob_1$

$$Prob_1 = \frac{Entry_P}{Entry_B} = \frac{2^{13} \times 7}{2^{14} \times 7} = \frac{1}{2}$$

- パターンでない文字列がパターンとみなされる SOG-like 法の照合誤りの発生確率 $Prob_{sog}$

$$Prob_{sog} = (Prob_1)^7 = \frac{1}{2^7}$$

ハッシュで生じる誤りは、次の通りである。

- あるパターンのハッシュ値と衝突する確率 $Prob_2$

$$Prob_2 = \frac{1}{2^{32}}$$

- ハッシュ値が衝突したパターンと，長さが一致する確率 $Prob_3$

$$Prob_3 = \frac{1}{256} = \frac{1}{2^8}$$

- あるパターンの長さとハッシュ値に衝突する照合誤りの発生確率 $Prob_{hash}$

$$Prob_{hash} = 8,192 \times (Prob_2 \times Prob_3) = \frac{2^{13}}{2^{40}} = \frac{1}{2^{27}}$$

提案手法全体で生じる誤りは、次のようになる。

- パターンでない文字列がパターンとみなされる提案手法の照合誤りの発生確率 $Prob_{app}$

$$Prob_{app} = Prob_{sog} \times Prob_{hash} = \frac{1}{2^{34}}$$

- テキスト 1 つあたりの照合誤りの発生確率 $Prob_{apt}$

$$Prob_{apt} = 2,048 \times Prob_{app} = \frac{2^{12}}{2^{34}} = \frac{1}{2^{22}}$$

- 全テキストあたりの照合誤りの発生確率 $Prob_{exp}$

$$Prob_{exp} = 16,384 \times Prob_{apt} = \frac{2^{14}}{2^{22}} = \frac{1}{2^8}$$

以上のように、誤りの発生は非常に低確率である。この結果は、帯域 100Mbps で利用率 64%のネットワークに IDS を設置した場合、およそ 24 日に 1 つ誤検知が発生する程度である。実際には、様々な値に偏りが生じるので、発生確率は上がると考えられる。しかし、この値のオーダーは IDS の誤検知率に比べると極めて小さいため、十分許容できると考えられる。

5 おわりに

本研究では、誤りの発生を許容する代わりに文字列探索を高速に行う手法を提案した。この手法で発生する誤りは、IDS の誤検知率と比べて非常に小さいため、運用上十分許容できると考えられる。

今後の課題としては、単純なアルゴリズムによってさらに多重化して処理効率を向上させることや、オープンソース IDS の Snort に実装し、実用上の有効性を評価することなどが挙げられる。また、単純な文字列探索だけでなく、メタキャラクタを用いたパターンマッチに対しても同様なアプローチを適用していきたい。

参考文献

[1] 淡路淳, 今泉貴史. IDS の誤検知除去に対するソフトウェア工学的アプローチ. 情報処理学会研究報告 (DSM — 分散システム/インターネット運用技術), Vol. 2007, No. 93, pp. 43–48, 2007.

[2] 小宅宏明, 宮地玲奈, 川口信隆, 重野寛, 岡田謙一. 機械学習によるネットワーク IDS の false positive 削減手法. 情報処理学会論文誌, Vol. 45, No. 8, pp. 2105–2112, 2004.

[3] Robert S. Boyer and J. Strother Moore. A fast string searching algorithm. *Communications of the ACM*, Vol. 20, No. 10, pp. 762–772, 1977.

[4] D. E. Knuth, J. H. Morris Jr., and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, Vol. 6, No. 1, pp. 323–350, 1977.

[5] R. M. Karp and M. O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, Vol. 31, No. 2, pp. 249–260, 1987.

[6] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, Vol. 18, No. 6, pp. 333–340, 1975.

[7] B. Commentz-Walter. A string matching algorithm fast on the average. In *Proceedings of the 6th International Colloquium on Automata, Languages and Programming*, No. 71 in LNCS, pp. 118–132, 1979.

[8] S. Wu and U. Manber. A fast algorithm for multi-patern searching. Technical Report TR-94-17, Department of Computer Science, University of Arizona, Tucson, AZ, 1994.

[9] E. P. Markatos, S. Antonatos, M. Polychronakis, and K. G. Anagnostakis. Exclusion-based signature matching for intrusion detection. In *Proceedings of the IASTED Internatinal Conference on Communicatons and Computer Networks (CCN)*, pp. 146–152, November 2002.

[10] Rong-Tai Liu, Nen-Fu Huang, Chih-Hao Chen, and Chia-Nan Kao. A fast string-matching algorithm for network processor-based intrusion detection system. *ACM Transactions on Embedded Computing Systems (TECS)*, Vol. 3, No. 3, pp. 614–633, August 2004.

[11] L. Tan and T. Sherwood. A high throughput string matching architecture for intrusion detection and prevention. In *32nd International Symposium on Computer Architecture (ISCA'05)*, pp. 112–122, June 2005.

[12] N. Jacob and C. Brodley. Offloading IDS computation to the GPU. In *22nd Annual Computer Security Applications Conference*, pp. 11–15, Miami Beach, Florida, December 2006.

[13] Sarang Dharmapurikar, Michael Attig, and John Lockwood. Design and implementation of a string matching system for network intrusion detection using FPGA-based Bloom filters. In *The 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, p. 22, April 2004.

[14] Leena Salmela, Jorma Tarhio, and Jari Kytöjoki. Multipattern string matching with q-grams. *ACM Journal of Experimental Algorithmics*, Vol. 11, p. 1.1, 2006.