ネットワーク侵入検知のための文字列照合のハードウェア設計

筒井　一雅†　　小柳　滋†
立命館大学大学院†

1．はじめに

近年のブロードバンドによる高速ネットワークの普及により、ネットワークを用いた不正侵入は増加傾向にあり、企業や国家機関だけでなく、家庭環境もが攻撃対象となっており、攻撃手法も多角化しつつある。こういった多様化する不正アクセスに対するセキュリティシステムとして NIDS が注目されている。ファイアーウォールと並行して利用することによりセキュリティ強化に貢献する重要な機能として、普及に向けた様々な研究開発が行われている。

一方で、ネットワーク上の情報トラフィックは増加の一途を辿り、処理作業が増加し、既存のソフトウェアによる処理では十分な処理速度が得られないという問題がある。

このような背景を踏まえ、本研究では NIDS をハードウェア設計することにより速度向上を狙ったシステムの構築を提案した。

2．侵入検知システム

2.1　NIDS

NIDS とは Network Intrusion Detection System の頭文字であり、ネットワークの通信を監視することで不正な侵入や攻撃を検知するシステムである。具体的には、ネットワーク上を流れるパケットを収集し、そのヘッダや各種のフラグやオプション、さらにはデータ部までチェックし解析する。そのためファイアーウォールでは見逃してしまうような巧妙に細工されたパケットも検出することができる。不正侵入を試みる際、不正な形式のパケットを送り付けたり、セキュリティホールが存在するサービスに関連するポートに接続したりするが、そういった疑わしいパケットを発見してこれを通知する。専用のマシンを増設するだけなので、導入と管理が容易であり、更に従来のネットワークシステムに影響を与えずに導入することが可能である。

2.2　Snort

Snort とは、Marty Roesch 氏によって 1998 年から開発されているソフトウェアベースのシグニチャ型 IDS であり、オープンソースのソフトウェアとしては最も広く使われている IDS である[2]。世界中のボランティアの手により様々な改良や検出ルールセットの作成が行われ、現在も日々進化し続けている。近年の不正アクセス事件の増加を受け、これら危険なアクセスを早期に検出することができる信頼性の高いソフトウェアとして、またルールセットのカスタマイズにより様々な運用ポリシーに適応できる柔軟性を持つソフトウェアとして多くの組織で導入されている。

2.3　NIDS と正規表現

正規表現は NIDS に広く利用されている技術であり、ネットワークにおける攻撃等の検出には欠かせないものとなっている。NIDS には多くの対攻撃パターンが必要であるが、正規表現を利用することでデータ量を軽減することが可能であり、増加傾向にある攻撃パターン数やデータトラフィックの急増に対応することができる。また、攻撃パターンは一般的に似たタイプのものが多く存在し、正規表現においては少しの変更で柔軟に対応できる場合が多く、日々増殖する新しいパターンに対しても効果的である。

3．ハードウェア設計

3.1　ハードウェア化の必要性

ネットワークの超高速化に対応する NIDS の実現は困難で、既存のソフトウェアによる処理では十分な処理速度が得られない状況である中、ハードウェアによる専用設計が解決策として注目されている。NIDS で重要な正規表現も、ハードウェアを用いた多くの研究がなされており、正規表現を使用できるパターンマッチングのハードウェア設計についての研究を開始した。

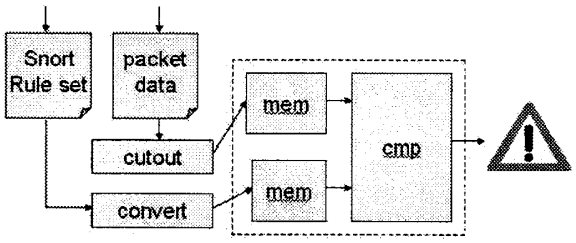

図 1　全体構成図

3.2　正規表現の実現手法

ハードウェアによる正規表現の扱い方は様々であるが、現在の主流となっている方法は NFA を用いたものであり、代表的なものには[1]などがある。これは、パターン全てに対する NFA の集合から回路を構成するという手法であり、正規表現を用いたパターンを動的に構成できるので柔軟な対応が可能であるが、ステートの量が膨大になりハードウェアの規模が大きくなることと、ステートの退

「Hardware design of string matching for Network Intrusion Detection System」
†「Kazumasa Tsutsui・Ritsumeikan Univ.」

避や復旧が困難であるという欠点がある。本研究では、これまでの研究で設計した文字列照合に正規表現を対応させるため、メモリベースの汎用マッチング回路を用いることで正規表現を実現する手法を提案した。

3.3 設計概要

全体の構成としては図1のようになる。まずSnortルール内におけるパターンマッチ部を抽出し、メモリに格納する。同じく通過パケットもメモリに格納し、各メモリから文字列照合器にデータを送ることでマッチング処理を行い、結果を出力する。今回ハードウェア設計したのはマッチング処理の部分のみであり、構成図を図2に示す。

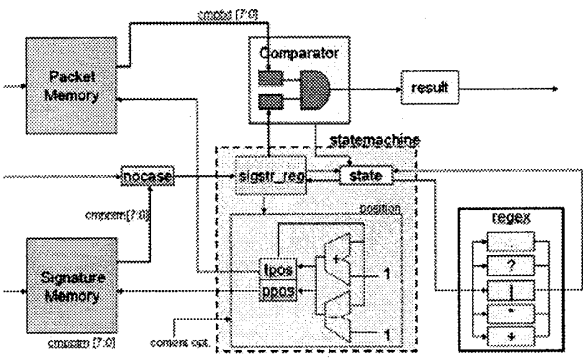

図 2 文字列照合のハードウェア構成

検索対象となる文字列を格納したシグニチャメモリから読み込まれた文字は、通常文字かメタ文字かを確認するためにまずステートマシンに送られる。通常文字であった場合はそのまま比較器に送り、パケットメモリから読み込んだ文字との比較を行う。メタ文字であった場合は、種類別に状態を遷移した後、Regexモジュールにおいてそれぞれの処理を行う。またステートマシンでは、比較器における文字の一致不一致や正規表現の有無などに伴い、それぞれメモリから読み込む文字の位置を示すポインタを持ち、比較処理を制御している。その他はSnort独自のルールである"nocase"や各種ルールオプションに対応するためのモジュールなどによって構成されている。"nocase"とは大文字と小文字の区別しないようにするための命令で、ルール本文に含まれていた場合はnocaseモジュールに信号を送り、シグニチャを読み込む際に変換を行う。文字を読み込み照合し、次のメモリアドレスを指定するまでの1文字分の照合は、全て2クロックで行われている。

4. 実験

4.1 並列化

スループット向上のため、ハードウェアに適した最も簡単な高速化手法である並列化を行い、それぞれ並列数ごとに処理速度と回路規模についての比較を行った。

4.2 実行環境

各メモリに格納するパケットデータやSnortルールの変換を行う部分が未完成のため、それぞれのデータは変換済みのものを予めメモリに格納して実験を行った。検索対象文字数1000文字、検索文字数4文字、検出箇所80箇所にて照合を行い、処理速度を求めた。シミュレータはXilinx社のModelSimXE III 6.2cを使用し、Familyを Spartan3E、Deviceに XC3S1600Eを用いた。並列数ごとのハードウェア概要と処理速度の結果を表1に示す。

表 1 ハードウェア概要と処理速度

並列数	Slice	処理時間 (Clock)	スループット (Mbps)
1	164	4393	151.6554
2	233	2198	303.1039
4	365	1089	611.7745
8	623	561	1187.5624
16	1174	275	2664.2466

5. 結果・考察

従来のソフトウェアによるSnortでの処理が、スループット1 Gbps以下と言われており、現在の設計アルゴリズムでは並列数8のものが匹敵していることが分かる。ただし、上記の実験結果はパターンマッチング処理のみの速度であり、パターンマッチ部に関係のないパケットヘッダ部を考慮すれば、実際のスループットはこれ以上のものになる。

また、今回設定したデバイスであるXC3S1600Eにおいて、16並列の文字列照合の回路規模はおよそ8%であり、概算すると200並列のものを実装でき、およそ30Gbpsの処理が期待できる。

ハードウェアの規模を抑えるために、回路構成の修正やアルゴリズムの改良が必要である。また、侵入・攻撃の手口に有用な処理に限定し高速化を狙うなど、応用に即した高速で小規模な正規表現の設計も行い、それぞれ評価したい。

6. おわりに

本研究では、NIDSに有用であるSnortルールセットや正規表現に対応したハードウェア設計によるメモリベースの文字列照合を提案した。

実験結果より実証された並列化の有用性より、更なる並列化が処理速度の向上へとつながることが推測できた。しかし、並列数を増大させることによる回路規模の増大は避けられず、最小限に抑えるためのアルゴリズムや回路構成の改良が必要となった。また、実際のパケットデータやSnortのルールセットを用いた実験を行うための変換部を設計することも今後の課題として残っており、処理内容に応じてソフトウェアとハードウェアを使い分けたシステムを構築したい。

参考文献

[1] Reetinder Sidhu, Viktor K. Prasanna, : Fast Regular Expression Matching using FPGAs",Proceedings of IEEE FCCM 2001, Apr 2001.

[2] Snort : open source network intrusion prevention and detection system utilizing a rule-driven language. http://www.snort.org/