УДК 004.274:004.056

А.Н. Давиденко, канд.техн.наук, ИПМЭ им. Г.Е. Пухова НАНУ, г. Киев
С.Я. Гильгурт, канд.техн.наук, ИПМЭ им. Г.Е. Пухова НАНУ, г. Киев

АЛГОРИТМЫ РАСПОЗНАВАНИЯ СТРОК В СИСТЕМАХ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ НА ПЛИС

The most known pattern-matching algorithms used in FPGA-based Network Intrusion Detection Systems are investigated. Their merits and disadvantages are exposed.

По мере увеличения числа и сложности сетевых компьютерных атак, а также из-за прекратившегося роста частоты микропроцессоров, становится все более проблематичной программная реализация сетевых систем обнаружения вторжений (ССОВ), соответствующий англоязычный термин – Network Intrusion Detection System (NIDS). Размеры баз данных сигнатур современных ССОВ исчисляются тысячами записей.

По этой причине в настоящее время существенно возрос интерес к реализации данных систем на базе программируемых интегральных схем (ПЛИС) типа Field Programmable Gate Array (FPGA) [1]. Высокая гибкость программируемой логики в сочетании с аппаратным ускорением позволяют эффективно использовать естественный параллелизм, присущий задаче распознавания строк, которая является наиболее ресурсоемкой операцией в современных системах обнаружения вторжений.

В настоящей статье рассмотрены основные алгоритмы, наиболее часто используемые в практических разработках ССОВ на базе ПЛИС.

Анализ информационных источников свидетельствует о наличии большого количества англоязычных публикаций по применению реконфигурируемых устройств на базе FPGA в составе NIDS. В то же время, информация на данную тему, фактически, отсутствует в литературе, издаваемой в странах СНГ.

Целью настоящей работы является исследование и оценка основных алгоритмов реализации процедуры распознавания сигнатур, используемых в системах обнаружения вторжений на базе программируемой логики по результатам анализа накопленного в мире опыта применения ПЛИС типа FPGA в составе NIDS.

Прежде всего, следует заметить, что исследуемая задача обладает высокой степенью параллелизма, причем, по двум направлениям: во-первых, несколько сетевых пакетов могут анализироваться одновременно; во вторых, сравнение может производиться сразу с несколькими сигнатурами. Решения на базе программируемой логики, аппаратные по своей природе, позволяют эффективно использовать этот параллелизм. В то же время, высокая гибкость и универсальность ПЛИС дают возможность воплощать в аппаратуре самые различные подходы и алгоритмы [2, 3].

Современные СБИС программируемой логики, содержащее миллионы эквивалентных логических элементов на одном кристалле, позволяют достичь скорости обработки сетевого потока в несколько Гбит/сек для баз данных емкостью порядка тысячи сигнатур, в то время как программная реализация систем обнаружения вторжений с аналогичной базой данных ограничивается скоростью в несколько сотен Мбит/сек [4, 5].

В работе [6] была упомянута еще одна специфическая особенность задачи распознавания строк, присущая ССОВ. Широкие возможности по оптимизации вычислительной структуры, реализующей процедуру распознавания предоставляет существенная информационная избыточность как входного потока данных, так и набора сигнатур. Способность эффективно реализовать эту особенность становится определяющим фактором при выборе конкретного решения.

1. Задача распознавания сигнатур.

Наиболее ответственным элементом любой системы обнаружения вторжений, использующей сигнатуры, является механизм распознавания строк (string matching). В потоке сетевого трафика алгоритм распознавания идентифицирует пакеты, содержимое которых соответствует сигнатурам известных атак. В случае выявления соответствия система обнаружения вторжений принимает соответствующие меры: пассивная ССОВ выдает сообщение об атаке, активная ССОВ (Intrusion Prevention System – IPS) фильтрует пакеты либо сбрасывает подозрительное сетевое соединение.

Одним из первых подходов к решению данной задачи было применение математического аппарата конечных цифровых автоматов (КЦА) [7, 8]. Громоздкость ЦА и высокая трудоемкость процедуры реорганизации цифрового автомата для реконфигурации ПЛИС, необходимость в которой возникает при каждом изменении списка сигнатур, существенно затрудняют практическое использование данного метода.

Чуть позже появились направления, связанные с использованием ассоциативной памяти (АП) [9, 10] и цифровых компараторов (ЦК) [11, 12]. Основанные на этих подходах решения обладают более высокой производительностью, но также не позволяют достичь быстродействия, достаточного для обнаружения вторжений на магистральных сетевых потоках в реальном масштабе времени. Кроме того, архитектуры, построенные с применением как АП, так и ЦК, требуют существенных вычислительных ресурсов ПЛИС.

Тем не менее, упомянутые выше решения оказались способными обеспечить, с одной стороны, большую гибкость по сравнению с фиксированными аппаратными устройствами на базе заказных СБИС, а с другой – намного более высокую производительность по сравнению с программной реализацией ССОВ.

Рассмотрим эти и некоторые другие наиболее часто упоминаемые в литературе алгоритмы решения задачи распознавания сигнатур в сетевых системах обнаружения вторжений на базе ПЛИС.

2. Методы аппаратного распознавания строк.

Существующие на сегодняшний день схемы аппаратного распознавания сигнатур можно классифицировать по методу, заложенному в их основу. Наиболее известны подходы, основанные на использовании:

- регулярных выражений (regular expressions);
- цифровых компараторов (discrete comparators);
- ассоциативной памяти (АП), в том числе предекодированной (CAMs / DCAMs);
- различных вариантов хеширования (hashing).

Рассмотрим каждое из упомянутых направлений подробнее.

2.1. Использование регулярных выражений. Цифровые автоматы.

Известно большое число разработок, связанных с аппаратным распознаванием строк, использующих для описания распознаваемых строковых шаблонов регулярные выражения. Элементарными элементами таких выражений являются символы, из которых состоят искомые строки, а также символы операций, таких как конкатенация – "()", или разделение – "|". Символ вопроса обозначает наличие либо отсутствие шаблона в строке, символ звездочки соответствует любому числу вхождений шаблона в сроку, включая нулевое. Например, выражение "a(b|c)?" обозначает одну из следующих комбинаций: "a", "ab" либо "ac". Соответствующие схемы можно найти, например, в работе [8]. Регулярные выражения генерируются для каждого шаблона в базе правил ССОВ, и по ним создается конечный цифровой автомат (КЦА) – Deterministic Finite Automata (DFA), либо недетерминистический цифровой автомат (НЦА) – Nondeterministic Finite Automata (NFA), который анализирует информацию последовательно, по одному байту, поступающему на его вход.

В указанной выше работе для создания цифровых схем регулярных выражений использовался Java-подобный язык описания аппаратуры, так называемый Java Hardware Description Language (JHDL).

Недостатком цифровых автоматов, как базиса для создания средств распознавания, является сложность их генерации. Время синтеза НЦА линейно возрастает с увеличением длины регулярного выражения, а затраты оборудования – пропорционально квадрату этой величины. Для КЦА обе эти зависимости экспоненциальные.

При необходимости добавления нового правила в базу данных сигнатур необходимо заново синтезировать ЦА. К тому же, и конечный, и недетерминистический цифровой автомат обрабатывает по одному байту за такт. Для повышения производительности может быть использован параллелизм на уровне сетевых пакетов [13]. При этом несколько экземпляров ЦА одновременно обрабатывают различные пакеты. В одной из ранних разработок по данному направлению НЦА синтезировался в микросхеме ПЛИС Virtex XCV100 производства фирмы Xilinx [7].

В проекте [8] также использовалась ПЛИС этой фирмы. Созданный

этими разработчиками генератор регулярных выражений на языке JHDL способен извлекать строки непосредственно из свободно распространяемой базы данных сигнатур Snort [14, 15] и синтезировать по ним загружаемую в ПЛИС конфигурацию.

В разработке, описанной в работе [13], для реализации распознавания строк на базе КЦА использовалась платформа программируемого пользователем расширителя порта – Field Programmable Port Extender (FPX).

2.2. Использование цифровых компараторов. Конвейерные вычисления.

Главным недостатком подхода, описанного в предыдущем подразделе, является ограничение по производительности – один символ за такт. Это привело к идее использовать при распознавании каждого правила в содержимом пакетов нескольких компараторов одновременно. Известны разработки с компараторами разрядностью в 32 бита и более.

Например, в работе [12] использовалось по четыре параллельных компаратора на правило. Каждое правило транслируется в структурный модуль на языке VHDL, который затем синтезируется и размещается в ПЛИС. Содержимое пакета подается на эти модули по 32-разрядной шине, таким образом, за каждый такт обрабатывается по четыре символа входной последовательности.

В проекте [11] задействовано одновременно N параллельных компараторов на правило. Дополнительно повысить тактовую частоту позволило применение конвейеризации. В качестве недостатков данного подхода следует отметить относительно высокие аппаратные затраты, требующиеся на синтез компараторов и большую задержку распространения, обусловленную длиной конвейера. Так, при реализации компараторов для распознавания тел всех правил из текущей на время публикации вышеуказанной работы базы данных сигнатур Snort потребовались три ПЛИС по 120000 эквивалентных логических элемента и еще одна – для распознавания заголовков правил.

2.3. Использование ассоциативной памяти.

Применение цифровые автоматов на ПЛИС позволило повысить производительность распознавания сигнатур примерно в 10 раз по сравнению с существовавшими на то время программными реализациями. С другой стороны, хранение правил в системе, построенной на базе ЦА, обходится дорого в смысле аппаратных затрат. При возрастании числа сигнатур в базе от сотен до тысяч записей внутренняя память ПЛИС становится неэффективной. Решить проблему недостатка памяти при сохранении достигнутой аппаратными решениями производительности позволили появившиеся чуть позже распознаватели шаблонов на базе АП. В таких устройствах в ассоциативной памяти предварительно инициируется несколько таблиц, а именно: таблицы протоколов, исходных и целевых IP-адресов, и содержимого строк. Каждый пакет снабжается вектором соответсвия, указывающим, какому из входов АП соответствует пакет. Процессор правил принимает пакет, по вектору соответствия соотносит с

внутренней базой правил и предопределяет действие для каждого правила.

В работе [10] применение ассоциативной памяти позволило вместить базу правил Snort в одну микросхему Xilinx Virtex XCV1000E.

Авторы разработки [9] дополнили цифровые компараторы предекодированной ассоциативной памятью. Данный подход позволил достичь производительности, соизмеримой с цифровыми компараторами при более низких аппаратных затратах. Недостатком решения является относительно высокая стоимость ассоциативной памяти, а также повышенное энергопотребление.

2.4. Использование функции хеширования.

В работе [16] для выполнения распознавания строк был использован фильтр Блума. Перед использованием такого фильтра для каждой сигнатуры из базы данных вычисляется k хеш-функций. Полученные числа определяют местоположение ячеек в битовом массиве размерностью m, которые устанавливаются в "1". Поступающие на вход распознающей структуры пакеты обрабатываются теми же самыми хеш-функциями, полученные значения сравниваются с сохраненными в массиве битами. Совпадение всех единиц означает, что анализируемый пакет с большой вероятностью содержит искомую сигнатуру. Для окончательного подтверждения гипотезы о совпадении необходимо выполнить дополнительную проверку обычным посимвольным сравнением. Если же хотя бы один бит не совпадет, это со 100%-ой вероятностью означает, что в данном пакете отсутствуют признаки данной атаки.

В связи с тем, что при заполнении фильтра многие сигнатуры взводят уже установленные в "1" биты, общий объем хранимой в устройстве информации о наборе правил существенно снижается по сравнению с другими подходами.

В упомянутой выше разработке [16] используется модификация фильтра Блума, при которой биты в массиве заменяются счетчиками. Добавление информации о новой сигнатуре в базе данных увеличивает на единицу значения соответствующих счетчиков. Удаление сигнатуры из базы уменьшает их значения на единицу, вплоть до обнуления. Такой прием позволяет поддерживать фильтр Блума в актуальном состоянии при редактировании записей в базе атак без необходимости полной повторной инициализации.

Чем больше размерность m битового массива по сравнению с числом сигнатур, тем лучше фильтр Блума представляет набор правил, и тем лучше работает механизм распознавания. Увеличение числа хеш-функций k также положительно влияет на качество функционирования фильтра. Однако, слишком большие значения указанных параметров могут привести к необоснованным затратам.

По сравнению с другими решениями фильтр Блума требует меньше памяти, проще перенастраивается и обеспечивает более высокую производительность, чем реализация на цифровых автоматах. Данный подход не содержит сам по себе ресурсоемкой операции посимвольного сравнения, но

лишь до тех пор, пока не происходит ложного срабатывания, вероятность которого тем выше, чем меньше параметры m и k по отношению к объему базы данных сигнатур.

Выводы по результатам настоящей работы можно сформулировать следующим образом.

В работе рассмотрены наиболее часто упоминаемые в литературе методы и алгоритмы аппаратного ускорения на базе ПЛИС операции распознавания строк в непрерывном потоке сетевых пакетов, применяемые в основанных на сигнатурах сетевых СОВ. Изложены основные принципы, проанализированы преимущества и недостатки каждого из подходов, приведены ссылки на конкретные разработки.

Обобщая изложенную информацию, следует обратить внимание на такой факт. Наличие относительно большого числа очень различных по своей сути направлений исследований, конкурирующих в течение нескольких лет, которое не привело к выявлению лидирующего метода, который бы значительно опережал другие подходы по основным показателям, свидетельствует о том, что в данной области не найдено однозначно превалирующего решения, которое могло бы претендовать на законченность и полноту.

Каждое из направлений имеет как некоторые преимущества перед другими, так и недостатки. Так, цифровые автоматы, синтезированные в ПЛИС, не обеспечивают высокую пропускную способность, сложны в построении и конфигурировании. Параллельные компараторы при большей производительности приводят к повышенным затратам оборудования и плохой масштабируемости. Решения, основанные на ассоциативной памяти, менее требовательны к ПЛИС, чем цифровые компараторы при соизмеримой производительности, но дороже и потребляют больше энергии. Наконец, фильтр Блума и сжатие базы сигнатур функциями кэширования позволяют уменьшить число сравнений, но обеспечивают лишь вероятностное распознавание, что требует дополнительных затрат на доуточнение результатов совпадения.

Следовательно, сформулированную выше техническую задачу распознавания сигнатур в реальном масштабе времени на реальных объемах базы правил современных сетевых систем обнаружения вторжений пока следует считать нерешенной.

1. Реконфигурируемые вычислительные системы: Основы и приложения. / *А.В. Палагин, В.Н. Опанасенко.* – К.: «Просвіта», 2006. – 280 с.

2. *Jiang W., Prasanna V.* Scalable Multi-Pipeline Architecture for High Performance Multi-Pattern String Matching // IEEE International Parallel and Distributed Processing

Symposium (IPDPS '10), April 2010.
3. *Monther A.* Configurable string matching hardware for speeding up intrusion detection. – USA. – 2006. – Ph.D. thesis.
4. *Proudfoot R., Kent K., Aubanel E., Chen N.* Flexible Software-Hardware Network Intrusion Detection System // Proc. 19th IEEE/IFIP International Symposium on Rapid System Prototyping, 2008, P. 182-188.
5. *Sourdis I., Pnevmatikatos D., Vassiliadis S.* Scalable multigigabit pattern matching for packet inspection // Proc. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.16 n.2, P. 156-166, February 2008.
6. *Гильгурт С.Я.* Анализ применения реконфигурируемых устройств в системах обнаружения вторжений // Тез. доп. Міжнар. наук.-техн. конф. «Моделювання-2010». Т.1. – Київ: Інститут проблем моделювання в енергетиці ім. Г.Є.Пухова НАН України, 2010. – С. 260-267.
7. *Sidhu R., Prasanna V.* Fast regular expression matching using FPGAs // IEEE Symposium on Field Programmable Custom Computing Machines (FCCM01), April 2001.
8. *Carver D., Franklin R., Hutchings B.* Assisting network intrusion detection with reconfigurable hardware // IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM02), April 2002.
9. *Sourdis I., Pnevmatikatos D.* Pre-decoded CAMs for Efficient and High-Speed NIDS Pattern Matching // Proceedings of 12th IEEE Symposium on Field Programmable Custom Computing Machines (FCCM04), April 2004.
10. *Gokhale M., Dubois D., Dubois A., Boorman M. etc.* Granidt: Towards gigabit rate network intrusion detection technology // Proceedings of the 12th International Conference on Field-Programmable Logic and Applications, Sept. 2002.
11. *Sourdis I., Pnevmatikatos D.* Fast, large-scale string match for a network intrusion detection system // Proceedings of 13th International Conference on Field Programmable Logic and Applications, 2003.
12. *Cho Y., Navab S., Mangione-Smith W.* Specialized Hardware for Deep Network Packet Filtering // Proceedings12th International Conference on Field-Programmable Logic and Applications, Sept. 2002.
13. *Moscola J., Lockwood J., Loui R., Pachos M.* Implementation of a content scanning module for an internet firewall // Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines. Napa, CA, 2003.
14. The Open Source Network Intrusion. http://www.snort.org
15. *Roesch M.* Snort: Lightweight Intrusion Detection for Networks // Proceedings of the 13th Conference on Systems Administration (LISA-99), Seattle, WA, November 7-12, 1999.
16. *Dharmapurikar S., Attig M., Lockwood J.* Design and Implementation of a String Matching System for Network Intrusion Detection using FPGA-based Bloom Filters // The 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '04), April 2004.

Поступила 18.10.2010р.