УДК 004.056:004.274

С.Я. Гильгурт, Б.В.Дурняк, Ю.М. Коростиль

ПРОТИВОДЕЙСТВИЕ АТАКАМ АЛГОРИТМИЧЕСКОЙ СЛОЖНОСТИ НА СИСТЕМЫ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ

Abstract. Some questions concerning the algorithmic complexity attacks against a network intrusion detection system (NIDS) are investigated. Namely, the main principles of so-called backtracking attacks as well as a solution that successfully thwarts such attacks are discussed. The general conclusion that only hardware realization based on FPGA is able to principally eliminate the threat of algorithmic complexity attacks against NIDS is made.

Введение. Сетевые системы обнаружения вторжений (ССОВ), соответствующий англоязычный термин – Network Intrusion Detection System (NIDS), являются важным инструментом в современной индустрии информационной безопасности. В постоянном противоборстве нарушитель / защита, имеющем место в данной сфере, ССОВ также вынуждены принимать активное участие. По этой причине актуальны вопросы противодействия различным приемам и методикам обхода защитных функций систем обнаружения вторжений злоумышленниками.

В связи с тем, что большинство современных ССОВ являются программными решениями, в мировой литературе накоплен определенный багаж знаний по методам обхода защиты и соответствующим контрмерам, в первую очередь, применительно к таким системам. Однако, из-за значительного увеличения объемов данных, передаваемых в сети, и недавней остановки роста частоты микропроцессоров в связи с достигнутым в микроэлектронике технологическим пределом, программные реализации все хуже справляются с поставленными задачами [1]. Данную тенденцию активно используют злоумышленники. В последнее время появляется все больше способов нейтрализации СОВ, направленных на их избыточную загрузку вычислительной работой, приводящую либо к сбоям в функционировании защищаемых информационных объектов, либо к недопустимому снижению их производительности.

Анализ последних исследований и публикаций свидетельствует о наличии большого числа исследований по вопросам обхода защиты, осуществляемой системами обнаружения вторжений, реализованными программно. Однако, в большинстве случаев для решения проблемы предлагаются программные же средства, что в ряде случаев не позволяет получить приемлемый результат.

Целью настоящей работы является рассмотрение возможностей существующих программных средств обнаружения вторжений по противодействию известным атакам на примере атак алгоритмической сложности, а также поиск путей преодоления выявленных трудностей и ограничений, в том

числе, с применением аппаратного ускорения на базе реконфигурируемых вычислителей.

1. Общие сведения об атаках алгоритмической сложности. К современным ССОВ предъявляются высокие требования по быстродействию. Неспособность обрабатывать сетевые пакеты в темпе их поступления может привести к негативным последствиям: в лучшем случае часть непроверенных пакетов будет проникать в защищаемую сеть, делая бессмысленной работу используемого защитного средства; в худшем – в системах предупреждения вторжений (Intrusion Prevention System – IPS), которые не только выявляют, но и нейтрализуют злонамеренный трафик, будет частично теряться полезная информация, передаваемая в сети.

Чтобы воспользоваться данным слабым местом систем обнаружения вторжений были разработаны так называемые атаки алгоритмической сложности (algorithmic complexity attacks) [2]. Вторжения данного класса основываются на том факте, что скоростные характеристики ССОВ, в первую очередь, программно реализованных, изменяются в широких пределах в зависимости от содержимого анализируемых сетевых пакетов.

В связи с тем, что большинство систем обнаружения вторжений, применяемых на практике в настоящее время, строятся на принципах сигнатурного анализа (signature-based IDS) [1], основная доля времени в их работе приходится на выполнение используемого алгоритма распознавания. Среди известного на сегодня большого количества алгоритмов распознавания подстроки в потоке символов, наибольшее распространение в СОВ получили те, которые имеют высокие скоростные показатели в среднем. Но именно такие алгоритмы, как правило, отличаются сильной зависимостью производительности от обрабатываемых данных [3]. Иными словами, для них возможно подобрать такой набор входных символов, при котором быстродействие снизится во много раз по сравнению со средним значением.

Типовая атака, использующая данную уязвимость, строится следующим образом. На предварительном этапе злоумышленник выясняет, какая ССОВ установлена на атакуемом объекте и какой алгоритм распознавания в ней используется. Затем злоумышленник формирует два вредоносных вектора: собственно последовательность команд или действий, направленных на достижение злонамеренной цели, и вектор прикрытия – последовательность символов для, воздействия на ССОВ с целью нарушения ее полноценного функционирования. На этапе реализации атаки из-за резкого повышения вычислительной трудоемкости алгоритма, выполняемого модулем распознавания, падает производительность атакуемой системы обнаружения вторжений, в результате чего большое количество сетевых пакетов не успевают проверяться. В их числе в защищаемую сеть с большой вероятностью проникают вредоносные пакеты, посылаемые нарушителем. Атака считается полностью успешной, если ни один из вредоносных векторов не будет распознан ССОВ.

2. Исследования атак на ССОВ и способов противодействия им. Проблемы атак на сетевые системы обнаружения вторжений широко исследованы в литературе. К наиболее ранним работам по данному направлению можно отнести [4] и [5].

В работах [6, 7] впервые рассмотрены атаки класса DOS (Denial Of Service - отказ в обслуживании), носящие характер атак аналитической сложности. В них используются уязвимости некоторых библиотек общего назначения, предназначенных для работы с регулярными выражениями и таблицами хеширования. Вынуждая подпрограммы многократно выполнять процедуры поиска с возвратом (бэктракинг – backtracking), злоумышленник навязывает системе намного более затратный по числу операций режим работы по сравнению со штатным случаем. В результате повышения вычислительной нагрузки на атакуемых объектах возникает ситуация отказа в обслуживании. Авторы показали пути противодействия данным атакам, основанные на модификации используемых алгоритмов и применяемых структур данных. В работе [8] более подробно рассмотрен класс атак, основанный на активном задействовании операций поиска с возвратом. Данная разновидность атак была названа бэктракинг-атаками и выделена как подкласс более общей группы атак алгоритмической сложности.

В работе [9] предлагается способ противодействия атакам, ориентированным на повышение вычислительной ресурсоемкости, реализованный на уровне системы. Подход основан на динамическом распределении нагрузки между несколькими модулями с целью обеспечения требуемой производительности. Регулирование при этом производится путем снижения приоритета вычислительных процедур; распределение вычислений между модулями происходит по принципу сброса нагрузки (load shedding).

Альтернативный вариант противодействия атакам алгоритмической сложности на ССОВ, предложенный в работе [10], заключается в распределении нагрузки между несколькими датчиками путем использования модуля-распределителя (для разделения потока данных) в комбинации с модулем-ограничителем и модулем-реассемблером (для восстановления логической целостности последовательности сетевых пакетов).

Известны также подходы, основанные на распределении функций датчика ССОВ между несколькими процессорами либо ядрами по принципам репликации [11, 12].

Но, как показывает анализ перечисленных выше работ, решения, основанные на использовании избыточных вычислительных ресурсов, удовлетворительно справляясь с повышенными сетевыми нагрузками, оказываются малоэффективными в борьбе с атаками алгоритмической сложности [8].

В ряде публикаций с целью предотвращения катастрофических последствий атак алгоритмической сложности предлагается контролировать процесс использования вычислительных ресурсов системой обнаружения вторжений. В упомянутой выше работе [5] специальная программа-монитор анализирует текущий режим работы процессора, а также историю его

загрузки. В случае обнаружения ненормальной длительности вычислительного процесса, программа, реализующая функции ССОВ, просто перезапускается. В работе [9] также используется мониторинг текущей загрузки; в случае обнаружения аномалий в работе системы, текущая операция прерывается, либо ее выполнение откладывается с более низким приоритетом.

Как можно видеть, подобные решения, основанные на мониторинге загрузки процессорных мощностей представляют собой удобный механизм, позволяющий обеспечить некоторый минимум гарантированного уровня производительности, жертвуя при этом в некоторых пределах достоверностью распознавания. Как следствие, в работе системы обнаружения вторжений эпизодически возникают перебои, что приводит к снижению надежности защиты.

Рассмотрены в литературе и другие аспекты данной проблемы. В частности, в работе [13] показано, каким образом можно противостоять атакам, использующим неоднозначности в описаниях используемых сетевых протоколов. В качестве средства противодействия предлагается выполнение операции нормализации сетевого трафика в комбинации с процедурой логически полного (stateful) анализа сетевых пакетов с учетом контекста [14] В работе [15] для аналогичных целей предлагается создавать специальную структуру данных для хранения сетевых атрибутов, таких как счетчик хопов (прямых соединений между хост-компьютерами в сети) между защищаемым компьютером и датчиком ССОВ. Показано, что подобное решение по эффективности равноценно нормализации, но при этом избавлено от риска нарушения семантической целостности передаваемых данных. Следует заметить, что упомянутые в последних работах аспекты противодействия атакам на ССОВ выходят за рамки настоящего исследования и являются отдельным направлением, не зависящим от его основной темы.

3. Бэктракинг-атака. В работе [8] исследована одна из разновидностей атак класса алгоритмической сложности, так называемая бэктракинг-атака, действие которой основано на использовании "наихудшего случая" в поведении алгоритма распознавания ССОВ.

3.1. Принцип действия атаки. Данная атака благодаря специальным образом подобранному содержимому сетевых пакетов вынуждает модуль распознавания многократно выполнять операцию поиска с возвратом (backtracking), что приводит к снижению скорости обработки пакетов модулем распознавания в некоторых случаях в полтора миллиона (!) раз по сравнению со средним значением. В результате проведения многочисленных экспериментов в данной работе показано, что под прикрытием бэктракинг-атаки могут быть осуществлены сотни вторжений в защищаемую системой ССОВ подсеть. При этом затраты трафика на засылку злонамеренного вектора прикрытия являются весьма скромными: отправки единственного сетевого пакета один раз в три секунды достаточно для полной нейтрализации системы обнаружения вторжений.

3.2. Пример описания правила ССОВ для бэктракинг-атаки. Рассмотрим, как работает атака на примере самой распространенной системы ССОВ с открытым исходным кодом Snort [16 – 18].

Синтаксически сигнатура базы данных Snort представляет собой набор из семи обязательных атрибутов, после которых в круглых скобках через точку с запятой следует в произвольном порядке перечень необязательных опций [18]. Правило срабатывает только когда выполняются все условия, задаваемые как основным атрибутами, так и опциями.

На рис. 1 в качестве примера приведено (с некоторыми упрощениями) правило, взятое из реальной базы данных Snort, направленное на обнаружение атаки «AudioPlayer jukebox exploit».

alert tcp $EXT NET any -> $HOME NET 99 (msg:"AudioPlayer jukebox exploit"; content:"fmt="; pcre:"/ ^(mp3|ogg)/",relative; content:"player="; pcre:"/.exe|.com/",relative; content:"overflow", relative; sid:5678)

Рис. 1. Пример правила Snort для анализа бэктракинг-атаки

Здесь: **alert** – атрибут типа, предписывающий системе в случае срабатывания правила записать пакет в журнал регистрации и выдать предупреждение;

tcp – атрибут протокола, который определяет принадлежность пакета к конкретному сетевому протоколу;

$EXT NET и **$HOME NET** – атрибуты-переменные, задающие IP-адреса источника и приемника сетевого пакета соответственно;

any и **99** – атрибуты, задающие номера портов (либо диапазоны портов) источника и приемника пакета соответственно;

-> – атрибут направления, который в данном случае предписывает анализировать только пакеты, передаваемые от сетевого узла-источника к узлу-приемнику.

Параметр ***msg*** в списке опций содержит название угрозы – «AudioPlayer jukebox exploit». Параметр ***sid*** указывает на внутренний номер данного правила в базе данных Snort.

Остальные параметры (в списке опций) относятся к распознаванию строк и указывают системе, какие последовательности символов необходимо искать в содержимом тела пакета, а также на взаимозависимости между ними. Так, параметр ***content*** явно задает искомую подстроку, а параметр ***pcre*** содержит описание регулярного выражения в формате, совместимом с языком Perl (широко используемом в UNIX-подобных ОС) – Perl Compatible Regular Expressions (PCRE) [19]. Параметр ***relative*** означает, что искомую подстроку следует искать не ранее позиции, в которой обнаружена подстрока, заданная предыдущим параметром.

Назовем шаблонами параметры, задающие в рассматриваемом примере подстроки поиска, и обозначим их следующим образом для удобства

дальнейшего анализа:

P1 = « content:"fmt=" »;

P2 = « pcre:"/^(mp3|ogg)/",relative »;

P3 = « content:"player=" »;

P4 = « pcre:"/.exe|.com/",relative »;

P5 = « content:"overflow",relative ».

На рис. 2 приведен пример последовательности символов в теле сетевого пакета, анализируемого системой обнаружения вторжений. На рисунке указан также номер каждого символа от начала строки.

```
fmt=aac_player=play_000_fmt=mp3_rate=14kbps_player=cmd.exe?overflow
01234567890123456789012345678901234567890123456789012345678901234567
          1         2         3         4         5         6
```

Рис. 2. Пример последовательности символов, распознаваемой ССОВ

На рис. 3 приведен результат работы алгоритма распознавания, используемого в системе Snort, представленный в виде содержимого стека программного модуля, реализующего данный алгоритм. Параметры в скобках указывают соответственно: имя шаблона, проверяемого на наличие в анализируемой строке; позицию в строке, на которой обнаружено начало шаблона; позицию в строке, на которой успешно завершено распознавание шаблона либо символ «F» в случае неуспешного завершения распознавания.

```
(P1, 0, 4) (P1, 0, 4) (P1,24,28) (P1,24,28) (P1,24,28) (P1,24,28) (P1,24,28)
           (P2, 4, F)             (P2,28,31) (P2,28,31) (P2,28,31) (P2,28,31)
                                             (P3,31,51) (P3,31,51) (P3,31,51)
                                                        (P4,51,59) (P4,51,59)
                                                                   (P5,59,67)
```

Рис. 3. Содержимое стека программного модуля распознавания

Как видно из рисунка, в позиции 67 анализируемой строки обнаружено выполнение условия распознавания, заданного всеми шаблонами из списка опций правила ССОВ.

Рассмотрим теперь, каким образом злоумышленник может создать злонамеренный вектор прикрытия, который вынудит модуль распознавания выполнять намного больше операций по сравнению со средним случаем.

Для правила, приведенного на рис. 1 шаблон P2 будет анализироваться столько раз, сколько раз встретится шаблон P1. (Последовательность символов «/^» в начале шаблона P2 означает, что он должен следовать непосредственно за шаблоном P1) Обозначим это число как n_1. Если за подстрокой P1 каждый раз будет следовать последовательность символов «mp3» либо «ogg», шаблон P2 будет распознаваться успешно, и алгоритм каждый раз будет вынужден начинать поиск шаблона P3. Если после этого шаблон P3 встретится n_2 раза, в каждом из $n_1 \cdot n_2$ случаев в свою очередь будет инициировано начало распознавания шаблона P4. Если далее в строке

встретится n_3 шаблонов P4, шаблон P5 будет искаться $n_1 \cdot n_2 \cdot n_3$ раза. На рис. 4 приведен пример последовательности символов, для которой $n_1 = n_2 = n_3 = 3$. При ее распознавании будет предпринято 27 попыток обнаружения шаблона P5 в исходной строке.

```
fmt=mp3fmt=mp3fmt=mp3player=player=player=.exe.exe.exe
012345678901234567890123456789012345678901234567890123456789012345678901234
          1         2         3         4         5
```

Рис. 4. Пример последовательности символов, распознаваемой ССОВ.

На рисунках с 5 по 9 приведены результаты работы алгоритма распознавания строки, приведенной на рис. 4, для первых пяти попыток, общее число которых будет составлять 27.

```
(P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4)
           (P2, 4, 7) (P2, 4, 7) (P2,28,31) (P2,28,31) (P2,28,31)
                                 (P3,21,28) (P3,21,28) (P3,21,28)
                                            (P4,42,46) (P4,42,46)
                                                       (P5,46, F)
```

Рис. 5. Первая попытка обнаружения шаблона P5

```
(P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4)
           (P2, 4, 7) (P2, 4, 7) (P2,28,31) (P2,28,31) (P2,28,31)
                                 (P3,21,28) (P3,21,28) (P3,21,28)
                                            (P4,46,50) (P4,46,50)
                                                       (P5,50, F)
```

Рис. 6. Вторая попытка обнаружения шаблона P5

```
(P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4)
           (P2, 4, 7) (P2, 4, 7) (P2,28,31) (P2,28,31) (P2,28,31)
                                 (P3,21,28) (P3,21,28) (P3,21,28)
                                            (P4,50,54) (P4,50,54)
                                                       (P5,54, F)
```

Рис. 7. Третья попытка обнаружения шаблона P5

```
(P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4)
           (P2, 4, 7) (P2, 4, 7) (P2,28,31) (P2,28,31) (P2,28,31)
                                 (P3,28,35) (P3,28,35) (P3,28,35)
                                            (P4,42,46) (P4,42,46)
                                                       (P5,46, F)
```

Рис. 8. Четвертая попытка обнаружения шаблона P5

```
(P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4) (P1, 0, 4)
           (P2, 4, 7) (P2, 4, 7) (P2,28,31) (P2,28,31) (P2,28,31)
                                 (P3,28,35) (P3,28,35) (P3,28,35)
                                            (P4,46,50) (P4,46,50)
                                                       (P5,50, F)
```

Рис. 9. Пятая попытка обнаружения шаблона P5

Как видим, даже относительно короткий вредоносный вектор, подобранный определенным образом, замедляет работу алгоритма более, чем на один десятичный порядок.

В общем случае для сетевого пакета длиной в m байт и правила, содержащего k взаимозависимых шаблонов, вычислительные затраты на распознавание каждого из которых составляют O(m) некоторых операций, злоумышленник может составить такой вредоносный вектор прикрытия, который вынудит модуль распознавания ССОВ выполнять $O(m^k)$ вычислительных операций [8].

Анализ базы данных сигнатур системы Snort, проведенный в исследовании [8], показал, что из примерно четырех тысяч правил около 8% являются в той или иной степени уязвимыми для бэктракинг-атак.

3.3. Противодействие. В упомянутой выше работе [8] предложен способ противодействия бэктракинг-атаке, основанный на концепции мемоизации (memoization). Основная идея состоит в следующем. Поскольку данная атака использует необходимость в процессе отработки алгоритма распознавания многократно возобновлять поиск сигнатур для каждого результативного смещения, во избежание излишних вычислений предлагается сохранять промежуточные состояния в специальной структуре, так называемой таблице мемоизации [20, 21]. В результате используемый алгоритм модифицируется таким образом, что разница в производительности для случая «в среднем» и наихудшего случая существенно сокращается, не затрагивая основной функциональности.

Как показали эксперименты, предложенное решение уменьшает снижение производительности при обработке злонамеренного трафика, содержащего атакующие пакеты, до соотношения не хуже, чем на один порядок по сравнению со штатным режимом работы. В большинстве же случаев процесс распознавания замедляется всего в два раза.

Следует отметить, что рассмотренное решение обеспечивает семантически строгое распознавание без снижения надежности обнаружения злонамеренного трафика в пользу сохранения производительности, в отличие от подходов на основе использования мониторинга текущей загрузки, упомянутых в разделе 2. Отметим также, что данная идея была доведена до практического использования в системе ССОВ Snort [16].

3.4. Аппаратные решения. Известно большое число работ по использованию аппаратных реконфигурируемых решений на базе программируемых интегральных логических интегральных схем (ПЛИС) для построения высокоскоростных средств распознавания, используемых, в том числе, в системах обнаружения вторжений [22 – 26]. В связи с тем, что по организации вычислительных процессов такие устройства принципиально отличаются от программных решений на базе традиционных процессорных систем, против них не может быть применена проанализированная выше бэктракинг-атака.

Выводы. В настоящем исследовании рассмотрена одна из разновидностей атак алгоритмической сложности, а именно, бэктракинг-атака. Показано, что предпосылкой для проведения подобных атак является сильная зависимость быстродействия программной реализации алгоритма распознавания от состава обрабатываемой информации, которая путем подбора злоумышленником содержимого сетевых пакетов может быть доведена до разницы в несколько порядков. Рассмотрено решение, позволяющее сократить этот разрыв на величину от одного порядка до двукратной разницы.

Тем не менее, нет оснований утверждать, что помимо исследованного выше подкласса не существует иных способов существенного замедления работы модуля распознавания и реализации тем самым атак на ССОВ, построенных по некоторому иному принципу. Так, например, в работе [27] исследованы возможности реализации атак алгоритмической сложности на системы, использующие в модулях распознавания алгоритм Ахо-Корасик, быстродействие которого, теоретически, не зависит от содержимого обрабатываемых данных.

Следовательно, для гарантированного противодействия атакам алгоритмической сложности недостаточно выбора правильных алгоритмов распознавания сигнатур. Ключевым моментом является техническое средство их реализации. Один из путей достижения цели, как указывалось выше, состоит в использовании аппаратных решений, в том числе, на базе программируемой логики.

1. *Коростиль Ю.М., Гильгурт С.Я.* Принципы построения сетевых систем обнаружения вторжений на базе ПЛИС // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Київ, 2010. – Вип. 57. – С. 87–94.
2. *Crosby S.A., Wallach D.S.* Denial of service via algorithmic complexity attacks. In Usenix Security, Aug. 2003.
3. *Смит Б.* Методы и алгоритмы вычислений на строках. Теоретические основы регулярных вычислений / Пер. с англ. – М.: Вильямс, 2006. – 496 с.
4. *Ptacek T.H., Newsham T.N.* Insertion, evasion and denial of service: Eluding network intrusion detection. In Secure Networks, Inc., Jan. 1998.
5. *Paxson V.* Bro: a system for detecting network intruders in real-time. In Computer Networks (Amsterdam, Netherlands: 1999), volume 31, pages 2435–2463, 1999.
6. *Crosby S.A.* Denial of service through regular expressions. In Usenix Security work in progress report, Aug. 2003.
7. *Crosby S.A., Wallach D.S.* Denial of service via algorithmic complexity attacks. In Usenix Security, Aug. 2003.
8. *Smith R., Estan C, Jha S.* Backtracking Algorithmic Complexity Attacks Against a NIDS. In Proceedings of the 22-nd Annual Computer Security Applications Conference, 2006
9. *Lee W., Cabrera J., Thomas A., Balwalli N., Saluja S., Zhang Y.* Performance adaptation in real-time intrusion detection systems. In RAID, Zurich, Switzerland, Oct. 2002.
10. *Kruegel C., Valeur F., Vigna G., Kemmerer R.* Stateful Intrusion Detection for High-Speed Networks. In Proceedings of the IEEE Symposium on Security and Privacy, pages 285–293, Oakland, CA, May 2002. IEEE Press.

11. The Snort network intrusion detection system on the Intel ixp2400 network processor. Consystant White Paper, 2003.
12. *Vermeiren T., Borghs E., Haagdorens B.* Evaluation of software techniques for parallel packet processing on multicore processors. In IEEE Consumer Communications and Networking Conference, Jan. 2004
13. *Handley M., Paxson V., Kreibich C.* Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In Usenix Security, Aug. 2001.
14. *Гильгурт С.Я., Дурняк Б.В., Коростиль Ю.М.* Быстродействие задач информационной безопасности в иерархии сетевого оборудования // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Київ, 2012. – Вип. 64 – С. 3–9.
15. *Shankar U., Paxson V.* Active mapping: resisting NIDS evasion without altering traffic. In IEEE Symposium on Security and Privacy, pages 44–61, May 2003.
16. SNORT [Электронный ресурс]. – Version 2.9.3.1 – Режим доступа: http://www.snort.org. – Загл. с экрана. – (Дата обращения: 17.03.2014).
17. *Давиденко А.Н., Гильгурт С.Я., Сабат В.И.* Аппаратное ускорение алгоритмов сигнатурного обнаружения вторжений в открытой системе информационной безопасности Snort // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Київ, 2012. – Вип. 65. – С. 94–103.
18. *Коростиль Ю.М., Гильгурт С.Я., Назаренко О.М.* Анализ базы данных системы информационной безопасности Snort и вопросы быстродействия // Моделювання та інформаційні технології. Зб. наук. пр. ІПМЕ НАН України. – Київ, 2012. – Вип. 66. – С. 77–84.
19. PCRE – Perl Compatible Regular Expressions [Электронный ресурс]. – Режим доступа: http://www.pcre.org. – Загл. с экрана. – (Дата обращения: 17.03.2014).
20. *Cormen T.H., Leiserson C.E., Rivest R.L.* Introduction to Algorithms. MIT Press/McGraw-Hill, 1990.
21. *Reps T.* “Maximal-munch” tokenization in linear time. ACM Transactions on Programming Languages and Systems, 20(2):259–273, 1998.
22. *Antonatos S., Anagnostakis K.G., Markatos E.P.* Generating realistic workloads for network intrusion detection systems // SIGSOFT Softw. Eng. Notes, vol. 29, no. 1, pp. 207–215, 2004.
23. *Attig M., Lockwood J.W.* SIFT: Snort intrusion filter for TCP. In Hot Interconnects, Aug. 2005.
24. *Sourdis I., Pnevmatikatos D., Vassiliadis S.* Scalable multi-gigabit pattern matching for packet inspection // Proc. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v.16 n.2, pp. 156-166, February 2008.
25. *Jiang W., Prasanna V.K.* Field-split parallel architecture for high performance multi-match packet classification using FPGAs // Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures, Calgary, AB, Canada, August 2009.
26. *Jiang W., Yang Y.-H. E., Prasanna V.K.* Scalable Multi-Pipeline Architecture for High Performance Multi-Pattern String Matching // IEEE International Parallel and Distributed Processing Symposium (IPDPS '10), April 2010.
27. *Shenoy G.S., Tubella J., Gonzalez A.* Hardware/Software Mechanisms for Protecting an IDS against Algorithmic Complexity Attacks. In Proceedings of the 2012 International Workshop on Security and Trust of Distributed Networking Systems (STDN-2012), May 2012.

Поступила 3.02.2014р.