UDC 621.391

J. N. DAVIES[1], P. COMERFORD[1], V. GROUT[1], N. RVACHOVA[2], O. KORKH[2]

[1] ***Creative and Applied Research for the Digital Society (CARDS), Glyndŵr University, Wrexham, UK***
[2] ***Poltava National Technical University, Poltava, Ukraine***

AN INVESTIGATION INTO THE EFFECT OF RULE COMPLEXITY IN ACCESS CONTROL LIST

An Access Control List (ACL) is an ordered list of rules which specify the action to take for any packet which is tested and matched against it. The list is arranged in order of decreasing priority, therefore if a match is made on a particular rule the packet is either permitted or denied and no further rules are evaluated. When configuring firewall rules it is possible to specify varying levels of granularity when examining the fields of a packet header. The most basic form of checking is on the source Layer 3 address. However there are more complex forms of the rule which enables further fields to be checked. This paper investigates the effect on the performance of a router when using these complex rules. In particular it concentrates on the checking of the port number field in TCP/UDP. A specialized simulator was built to help understand the process undertaken by the router. There are results of the investigations and a recommendation on how to improve performance in certain areas.

Keywords*: IP packet filtering, ACL complexity, Network Performance, Delay through Routers, Access Control List, ACL optimization, ACL Simulator, Firewalls.*

Introduction

Infrastructure security within a domain is normally implemented in either firewalls or routers containing Access Control Lists (ACLs). An ACL is an ordered list of rules which accepts or rejects a packet based on one or some of its characteristics. Typically, a packet may be considered on the basis of its source, destination or protocol, although other features may be relevant [1]. Fig. 1 gives an example of a typical ACL in the syntax of the Cisco Internetwork Operating System (IOS). The use of the terms permit and deny reflect the original role of ACLs in passing or blocking traffic.

```
deny tcp 91.182.189.193 0.0.0.3 135.59.89.54 0.0.0.3 eq 80
permit ip 194.247.4.143 0.0.0.3 157.54.173.200 0.0.0.15 gt 1023
deny udp 98.99.186.58 0.0.0.7 202.86.156.57 0.0.0.63
deny udp 164.101.111.116 0.0.0.63 any eq 129
permit udp 4.92.186.57 0.0.0.15 110.165.54.241 0.0.0.63 eq 53
permit udp 199.150.89.32 0.0.0.15 177.148.27.103 0.0.0.15
permit udp any  48.184.250.133 0.0.0.15
permit ip 211.151.242.98 0.0.0.3 195.172.11.133 0.0.0.15
deny udp 43.15.152.198 0.0.0.7 148.144.102.233 0.0.0.3
permit ip any any
```

Fig. 1. Typical Domain allocation

Each packet to be tested against an ACL is compared with the first rule, then the second, and so on, until a rule matches its profile. The packet is then permitted or denied accordingly and no more rules are considered.

There is an implicit 'deny all' rule terminating each list to deal with packets not matched by any other rule. A precise treatment of rule and packet formats and profiles is given in [2]. This level of analysis is not required, here, except in its final formulation of the problem. However, it is necessary to note that rule order is critical in an ACL.

Rules are defined in a number of different formats depending on the hardware and operating system utilized e.g. Cisco IOS[2], Juniper Networks JUNOS (firewall filters) [3] or Linux (IPTables) [4] Fig. 1. Despite the differences in format the functionality is very similar; a packet is either forwarded or discarded based on the matching of fields in the packet with the specified rules. Rules are created from a permit or deny statement followed by variables which can include source address, destination address, protocol or port addressing within the packet. Since every packet has to be tested significant delays can result from the introduction of such techniques due to the filtering requirement [5].

```
Cisco Access Control List
access-list 101 deny tcp 192.168.1.10 host 10.0.0.1 host 23

Linux IPTables list
iptables -A INPUT -s 192.168.1.10 -d 10.0.0.1 -p tcp -dport
23 -j DROP

Junos Firewall filter
firewall {
  family inet{
    filter filter- 1 {
      term allow-webserver-connections {
             from {
                   destination-address {
                        192.168.1.10/32;
                                  }
                        protocol tcp;
                        destination-port [ http 80 ];
                                 }
              then  {
                    accept;
                          }
                           }
                            }
                                 }
              }
```

Fig. 2. Typical Domain allocation

Attempts have been made to use various techniques to optimize the delay through routers caused by ACLs [6]. Optimization of packet filtering performance has been the subject of intense research for the past decade [7]. A number of studies have identified rule relations within an ACL which may result in redundant or conflicting rules [8].

This paper investigates the significance of the delays encountered through the use of various ACL techniques. Factors which contribute to the delay incurred by packets passing through a router are identified and subsequently, a number of experiments were conducted to quantify these. Suggestions were made to help in the construction of ACLs to improve performance based on experimental results. Only delays through network equipment were considered in this paper.

1. Rule Complexity

When checking packet header fields for security purposes it is usually necessary to examine additional fields in the IP and transport layer headers. This type of ACL is known as extended by Cisco but similar filtering can be configured using IPTables or other manufacturer's format. As a minimum a rule in an extended ACL will allow the destination IP address of a packet to be examined using a single or group of addresses.

Table 1

IP Packet

Ethernet Header	IP Header	e.g. TCP Header	Data	Checksum
	1. Source Address 2. Destination Address 3. Protocol	1. Port Number		

Additionally, it is possible to test against the source address and/or source and/or destination port values and other fields such as flag bits in the TCP header and various ICMP message types. When performing tests on port values, it is possible to specify different operators depending on the tests to be made on a packet. Typically packet filtering schemes such as those implemented by Cisco will be able to check port values using the operators equal, not equal, greater than, less than or an arbitrary range of port values. An extended ACL in Cisco format is shown in the figure below using rules containing various types of operators, fig. 1.

1.1. Testing

An experimental test-bed was set up to perform tests using real world filtering hardware to determine if the results were consistent with those found when using a simulator. The network consisted of a single router with a source and destination host connected to either interface. The filtering rules were applied to the ingress port of the router and a packet generator was used on the source host to provide the appropriate packet types. Using ICMP packets was not suitable for this experiment as TCP/UDP packets were required with matching IP addresses and non-matching port numbers. Of course the final rule in the ACL will be the permit all which is the rule that will be matched. This was to ensure that at a minimum, the source and destination fields of the IP header would be checked.

The tests were performed using both a Cisco router and a Linux machine running IPTables. Cisco tests were conducted on the same router hardware using two versions of the Cisco IOS, basic and advanced functionality. Previous studies have shown that the processing time associated with a packet filter varied considerably depending on the IOS versions used [9].

Consistent equipment was used for all the experiments and a single monitoring machine to ensure that a synchronized clock is used for timing the delay. The number of rules in the ACL was kept to 100. To calculate the delay, the packets were captured on the monitoring station using Wireshark on a dual ported monitoring station. Timestamps are obtained at times T1 and T2 and the difference is used to determine the processing time for a particular ACL type. In addition, the experiment was repeated without any filtering applied to the interface so a benchmark could be obtained. The topology used in the experiment is shown in fig 3.

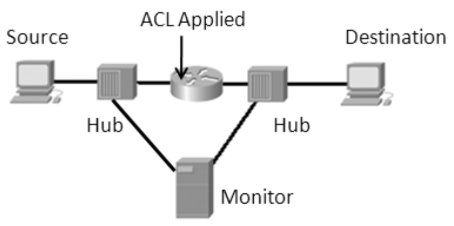

Fig. 3. Test Network

1.2. Delay caused by packet routing

When a packet is passed through a router the IP header has to be checked against the Routing Table to find out which port of the router the packet should be transmitted out to. This will happen whether an ACL has been used or not.

To get some idea of the effect of simple routing test where done with no ACL in a Cisco router and no IPTables configured on a Linux machine.

Fig. 4 shows the result of these tests. It can be seen that the Linux machine is much faster to carry out routing and the time taken is more predictable. The Cisco router with an advanced operating system was by far the slowest and there was quite a spread in the times observed.

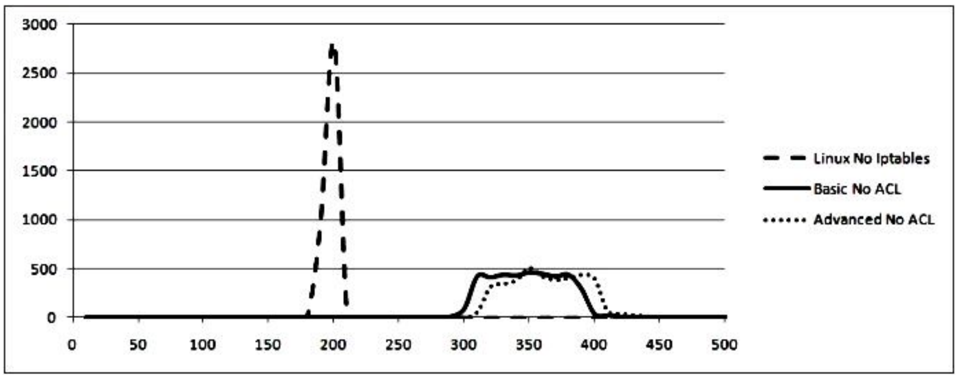

Fig. 4. Delay through router with different OS

2. Simulator

In previous work an attempt was made to use off the shelf network simulators e.g. ns 2-3, Opnet but for various reasons this was not a success [10]. For this paper a specialized simulator was written to investigate the action undertaken by a router. This was a simple program that used programming techniques to undertake the same functions as a router would take to interpret the ACL. This simulator was capable of loading up the same ACL that was loaded into the Cisco routers. A packet format was then defined which was the same as that used in iperf [11] for the real network tests and this was tested against the ACL to check the performance. Timestamps were used before the start of a rule and then again after the rule.

Due to the proprietary nature of Cisco equipment it was not possible to make modifications to the Cisco IOS to allow exact time measurements to be taken for each filtering rule. It was considered too time consuming to modify the code on open source packet filters such as IPTables. The Layer 4 protocol field is first examined to determine a match and subsequently, Layer 3 IP address fields are examined followed by source or destination port values if specified. Results from the simulation are useful as they serve as an indication of the expected results when a packet is checked against lists of rules with varying complexity. The rule checking process used in the simulator can be summarized using pseudo-code fig. 5. The process uses three helper functions: match protocol, match IP and match port to perform the checks for IP addresses, port values and protocol number values. Match protocol function is a simple comparison using the protocol number contained in the packet header with the protocol value of the rule. Match IP function takes three parameters, the source IP address of the packet and rule along with the mask for the rule specified using the Cisco wildcard format. A bitwise check of the packet address is carried out for only those bits which have a corresponding binary zero in the wildcard mask. Each bit in the address must be equal for a match to occur. Match port function also requires three parameters: the port value from the packet and rule and also the rule operator for the rule e.g. equal, greater than, less than since the operator will change the range of values to be tested. If a checking function such as match protocol returns a false value, i.e. there is no match, then the rest of the fields are not checked and the next rule is evaluated. The process continues for all other rules in the list until the last rule has been evaluated. At this point, if a match has not been found then the packet is denied by default.

```
Input: rules, packet
Output: result

for each rule in rules do
  if !MatchProtocol(packet.ProtocolNumber, rule.ProtocolNumber) then continue
  endif
    if !MatchIP(packet.SourceAddress, rule.SourceMask) then continue
    endif
      if !MatchIP(packet.DestAddress, rule.DestMask) then continue
      endif
        if !MatchPort(packet.SourcePort, rule.SourcePort, rule.Operator) then continue
        endif
          if !MatchPort(packet.DestPort, rule.DestPort, rule.Operator) then continue
          endif
result = rule.Type
endfor
result = Deny
```

Fig. 5. Pseudocode for rule checking process

2.1. IP Packet

The simplest packet to be tested by an extended ACL is an IP packet (fig. 6). This is because the processor has to check the protocol and the source and destination addresses only.

Results for the Linux machine show that a delay of around 200 μ secs is observed.

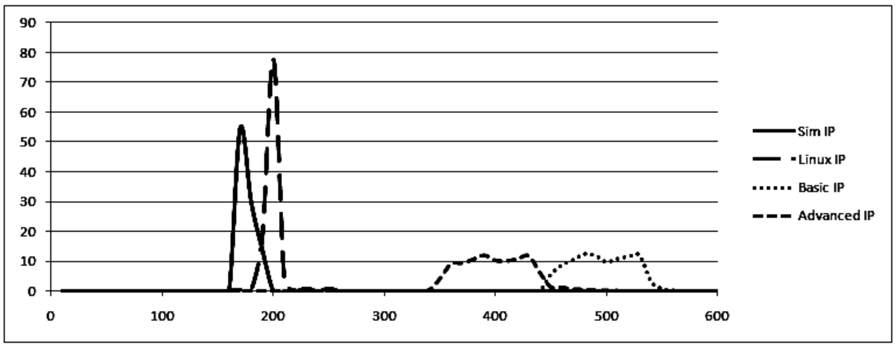

Fig. 6. Delay through router for IP Packet

However when the results were analyzed for the Cisco router the delay using a Basic OS was twice that of the Linux machine and using an Advanced OS even longer averaging around 500 μsecs. Additionally the variation in the delay is much larger in the Cisco router. The simulator created for this paper showed results that were remarkably close to the Linux machine.

2.2. Operators in ACL Rules

Further work was carried out to investigate the delay experienced by packets for UDP packets that forced a comparison to be carried out on the Port Number. A series of results was collected were the packets were matched against the operators equal, greater than, less than or an arbitrary range of port values. Additionally some OS support the ability to specify explicit Port Numbers Fig. 7.

The results obtained for the real equipment are very much as expected i.e. the delays for testing with operators equal, greater than, less than produce identical delays. Additionally the delays from an arbitrary range of port values show a longer delay as does using a number of explicit Port Numbers. A significant conclusion can be drawn from this, 5 individual rules checking for a Port Number would take around 7750 μsecs but explicitly specifying 5 Port Number in one rule would take about 1700 μsecs an improvement of 350%.

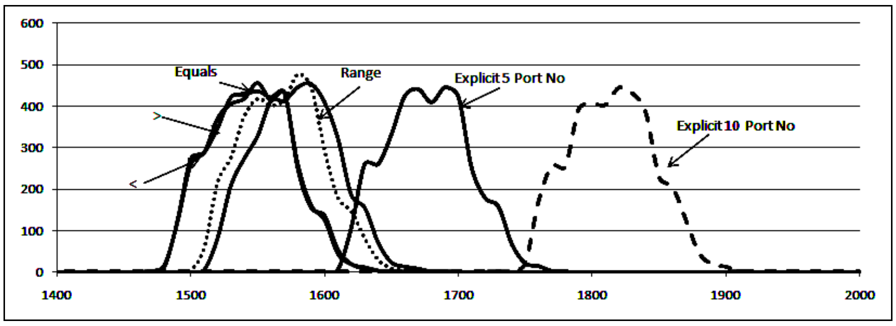

Fig. 7. Delay through router with Basic OS

2.3 Simulator results for Rule Delay

The results obtained from the simulator provide delay values for every rule that is being checked so to get results that are compatible with that from the real network is to just consider the time for the last rule i.e. 100th rule, fig. 8. It would be very easy to make the results obtained from the simulator to be the same as those obtained from the Linux machine since the difference is just a time difference, fig 9. This could be implemented by inserting a delay in the loop. To simulate the Cisco results is a bit more complex to achieve but it is still possible. Clearly a delay is required which is much greater than the delay for Linux but additionally a spread of the times needs to be catered for. This could be done by choosing a random number within a range ±100 μsecs of the value. The increase in the delay for a Basic OS would be of the order of 250% and for an Advanced OS 700%. The simulator would have an option to select for the type of OS and so this would then make the results more comparable.

Conclusion

When implementing security on routers it is important to consider the consequences of the type of rule invoked as well as the number of rules. Some rules are much more complex than others and can take longer to run resulting in a longer delay through the router.

To investigate this statement a series of tests were performed on a real network to prove the point and to quantify the values. Experiments were carried out to test rules using different operators i.e. equals, greater than, less than and range.

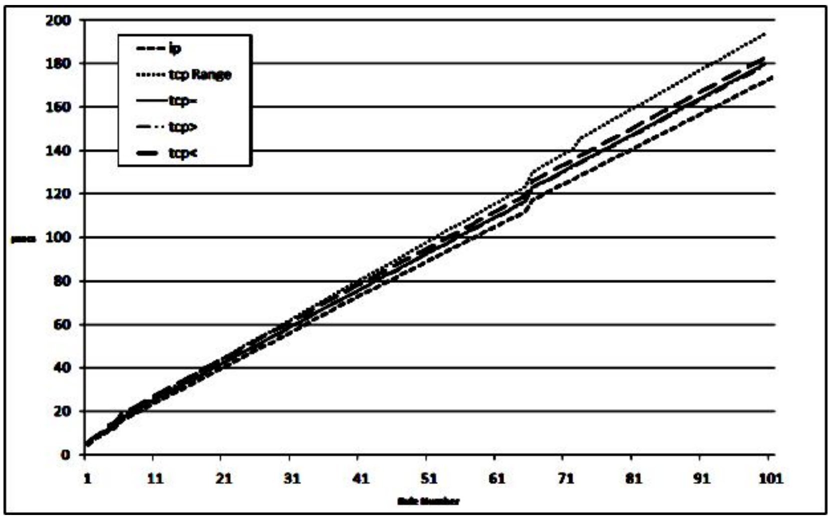

Fig. 8. Simulator Results for individual rules

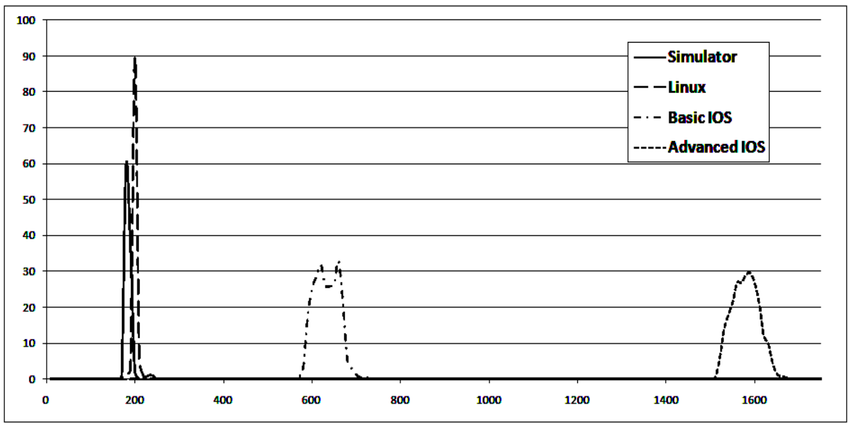

Fig. 9. Simulator Results compared to other OS

Additionally there are further parameters that can be specified that enable a number of explicit values to be checked. The testing was carried out with a Linux machine running Fedora and a Cisco router running 2 different versions of the OS. These were then compared with a specialized simulator to understand the process adopted within the Cisco OS.

The results showed that when one operator was used e.g. =, >, < then the results were consistent, when the range operator was used then the delay was larger and when a number of explicit Port numbers wee tested then the delay was again much greater.

However on important conclusion was that by using a number of explicit Port numbers it was much quicker than insert more rules. By explicitly adding 5 different Port Numbers to 1 rule rather than using 5 separate rules a saving of the order of 350% could be gained.

A series of test were undertaken to compare the results obtained by the specialized simulator and those measured in a real router. The results of the simulator were very similar to that found in the Linux tests. However there was quite a large discrepancy between these results and those found in Cisco routers. The spread of the times in the simulator and the Linux machine were very similar the spread being around 5% however the difference in the times between the simulator and the Cisco router with the Basic OS was250% and for the Advanced OS is around 700%.

It is intended to carryout further work to make the simulator more realistic, clearly this will have to have an option that allows the OS type to be selected.

References

1. Grout, V. Real-time optimization of Access Control Lists for efficient internet packet Filtering [Text] / V. Grout, J. McGinn, J.N. Davies // Journal of Heuristics. – October 2006. – V. 13, No. 5 – P. 435 - 454.

2. Sedayao, J. Cisco IOS Access Lists [Text] / J. Sedayao // O'Reilly & Associates. – Inc., Sebastopol, CA, USA, 2001. – P. 22 - 31.

3. Marschke, D. JUNOS Enterprise Routing [Text] / D. Marschke, H. Reynolds // O'Reilly Media Inc. – 2008.

4. Purdy, G. Linux Iptables [Text] / G. Purdy // O'Reilly Media Inc. – 2004.

5. Davies, J.N. Improving the Performance of IP Filtering using a Hybrid Approach to ACLs [Text] /

J.N. Davies, V. Grout, R. Picking // Proceedings of the 8th International Network Conference (INC2010). – Heidelberg, Germany, 6 - 8 July 2010.

6. Al-Shaer, E.S. Modelling and Management of Firewall Policies [Text] / E.S. Al-Shaer, H.H. Hamed // IEEE Transactions on Network and Service Management. – April 2004. – Vol.1, No. 1. – P. 2 – 10.

7. Hari, B. Detecting and Resolving Packet Filter Conflicts [Text] / B. Hari, S. Suri, G. Parulkar // Proceedings of the 19th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM00). – Tel Aviv, Israel, March 26 - 30, 2000. – V. 3. – P. 1203 - 1212.

8. User Guide for ACL Manager 1.5, Optimizing ACLs [Electronic resourse] / Cisco Systems. – Access to: http://www.cisco.com/en/US/products/sw/cscowork/ps2425/index.html. – 9.02.2012.

9. Davies, J.N. Optimization of Delays Experienced by Packets due to ACLs within a Domain [Text] / J.N. Davies, P. Comerford, V. Grout // Proceedings of ITA 11 Fourth International Conference on Internet Technologies & Applictions. – Glyndwr University, Wrexham, UK, September 6 – 9, 2011. – P. 341 - 348.

10. Davies, J.N. Eliminating dependencies in linear ACLs [Text] / J.N. Davies, P. Comerford, V. Grout // 7th Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2011). – Furtwangen, Germany, October 2011.

11. Iperf official site [Electronic resourse]. – Access to: http://iperf. sourceforge.net. – 9.02.2012.

Поступила в редакцию 25.03.2012

Рецензент: канд. техн. наук, доцент А.В. Горбенко, Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков, Украина.

ДОСЛІДЖЕННЯ ВПЛИВУ СКЛАДНОСТІ ПРАВИЛ В ACCESS CONTROL LIST

Д.Н. Девіс, П. Комерфорд, В. Граут, Н. Рвачова, О. Корх

Access Control List (ACL) представляє собою послідовність правил, які визначають дії над будь-яким пакетом, що надходить на маршрутизатор. На практиці використовуються складні форми конфігурації ACL, що передбачають перевірку додаткових полів заголовку пакету. В роботі досліджується вплив складності правил ACL на продуктивність маршрутизатора. Увага сконцентрована на перевірці номера порта TCP/UDP. З метою дослідження процесу перевірки правил ACL розроблено ACL simulator. В статті представлені результати досліджень та рекомендації по покращенню продуктивності маршрутизатора.

Ключові слова: фільтрація IP-пакетів, складність ACL, продуктивність мережі, затримка в маршрутизаторі, Access Control List, ACL оптимізація, ACL simulator

ИССЛЕДОВАНИЕ ВЛИЯНИЯ СЛОЖНОСТИ ПРАВИЛ В ACCESS CONTROL LIST

Д.Н. Девис, П. Комерфорд, В. Граут, Н. Рвачева, О. Корх

Access Control List (ACL) представляет собой последовательность правил, определяющих действия над любым пакетом, который приходит на маршрутизатор. На практике используются сложные формы конфигурации ACL, предполагающие проверку дополнительных полей заголовка пакета. В работе исследуется влияние сложности правил ACL на производительность маршрутизатора. Внимание сконцентрировано на проверке номера порта TCP/UDP. С целью исследование процесса проверки правил ACL разработан ACL simulator. В статье представлены результаты исследований и рекомендации по улучшению производительности маршрутизатора.

Ключевые слова: фильтрация IP-пакетов, сложность ACL, производительность сети, задержка в маршрутизаторе, Access Control List, ACL оптимизация, ACL simulator.

Девис Джон Н. – д-р техн. наук, профессор, Университет Глиндор, Рексем, Великобритания, e-mail: j.n.davies@glyndwr.ac.uk.

Комерфорд Пол – аспирант, Уиверситет Глиндор, Рексем, Великобритания, e-mail: p.comerford@glyndwr.ac.uk.

Граут Вик – д-р техн. наук, профессор, Университет Глиндор, Рексем, Великобритания, e-mail: v.grout@glyndwr.ac.uk.

Рвачева Наталия – канд. техн. наук, старший преподаватель кафедры компьютерной инженерии Полтавского национального технического университета имени Юрия Кондратюка, Полтава, Украина, e-mail: rvacheva_n@mail.ru.

Корх Олег – канд. техн. наук, старший преподаватель кафедры прикладной математики, информатики и математического моделирования Полтавского национального технического университета имени Юрия Кондратюка, Полтава, Украина, e-mail: korkholeh@gmail.com.