УДК [004.056](https://teacode.com/online/udc/00/004.056.html).5; 004.738.5

**РЕАЛИЗАЦИЯ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ ДЛЯ ВЫЯВЛЕНИЯ АТАК – ЧЕЛОВЕК ПО СЕРЕДИНЕ.**

***Степанов П.П.[[1]](#footnote-1), Никонова Г.В.[[2]](#footnote-2)***

***Цель исследования*** *заключается в разработке клиент-серверного приложения для выявления и оперативного реагирования на атаки типа “человек по середине’. Которое позволяло бы тонко настраивать политики безопасности для разных узлов сети. В статье описано разработка клиентского приложения этой утилиты.*

***Методом проведения*** *исследования является разработка сетевой утилиты, работающей как системная служба windows.*

***В результате исследования*** *была разработана утилита, реализованная как системная служба windows. Которая собирает текущую информацию узла сети. Данная утилита собирает такие параметры как адрес DHCP сервера, адреса DNS серверов, адрес шлюза полмолчания, IP адрес текущей машины, имя текущей машины ARP таблицу и таблицу маршрутизации. Которые будет передаваться для дальнейшего анализа и реагирования на инциденты ИБ.*

*Ключевые слова: ARP протокол, DHCP сервер, DNS сервер, ARP-spoofing, перехват трафика, windows service.*

**DOI:**10.21681/2311-3456-2024-1(2)-xx-yy

# Введение

Сетевые атаки типа «человек посередине» [MITM](https://ru.wikipedia.org/wiki/MITM) (англ. Man in the middle) являются серьезно проблемой информационной безопасности. Основная идея этих атак заключается в том, что трафик не идет напрямую между двумя узлами сети, а проходит через злоумышленника. Злоумышленник может просматривать и модифицировать проходящий через него трафик. Возникла идея в реализации клиент-серверной утилиты для выявления и оперативного реагирования на подобные атаки. Данная утилита должная состоять и клинского приложения (которое будет устанавливаться на все компьютеры) и серверного приложения которая будет анализировать поступающие данные. В статье описано написание клинского приложения под операционную систему Windows. Которое будет работать как системная служба и отправлять данные о настройках сети, которые могут быть компрометированы про проведение [MITM](https://ru.wikipedia.org/wiki/MITM) атак[3-5].

## Описание решения (Solution) NetworkMonitor

На данный (Solution) NetworkMonitor состит из 4 проектов:

1. NetworkMonitor.Common – Сборка содержит в себе базовые вещи, которые используются во всех других сборках (рисунок 1):

Constants - константы такие как сообщения оба ошибках, сообщения для логирования, значения по умолчанию.

Dto – (Data Transfer Object) - данные для передачи от клиента серверу.

Enums - содержит перечисления.

Exceptions – содержит исключения, которые могут возникать в ходе работы.

ExtensionMethods – содержит методы расширения для базовый классов .net, например для NetworkInterface.

Interfaces – содержит описание контрактов, которые необходимо реализовать для каждой платформы (Windows, Linux).

Settings - содержит классы с настройками.

1. NetworkMonitor.Implementation – Сборка содержит в себе реализации интерфейсов для различный платформ.
2. NetworkMonitor.Tests – Сборка содержит в себе Unit тесты для проверки корректной работы различный модулей. Папка Builders – содержи в себе реализации GoF паттерна Builder для создания сущностей, которые будут использовать в Unit тестах.
3. NetworkMonitor.WindowsService – Сборка содержит в себе клиент для системы Windows, которая может быть установлена в систему в качестве системной службы (Windows Services). Также здесь настраивается внедрение нужный зависимостей DI (Dependency injection) необходимых для работы именно под операционной системой Windows.

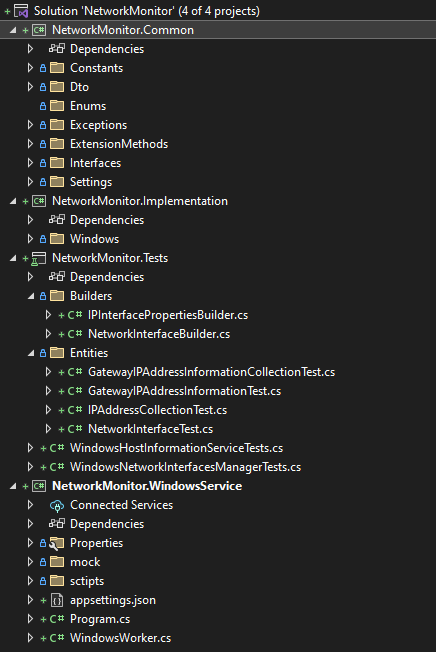


Рисунок 1 – Схема решения (Solution)

## **Описание интерфейсов и Dto.**

Прежде чем приступить к реализации функционала ПО на уровне классов и методов необходимо продумать функционал на максимально абстрактном уровне – уровне контрактов или интерфейсов (interface), прописав сигнатуры методов, которые обязан реализовывать класс. Это нужно для дальнейшего использования DI (Dependency injection) что уменьшает связность кода и делает его более гибким для изменений и более легким для написание модульных тестов (Unit).

* IHostInformationService – Интерфейс содержит в себе контракт на получения информации о узле сети (рисунок 2), и обязует класс содержать в себе следующие методы:
* GetDhcp() - получение IP адреса DHCP сервера. При проведении ряда [MITM](https://ru.wikipedia.org/wiki/MITM) таких как DHCP spoofing или Evil Twin он может быть изменен.
* GetGateway() - получение IP адреса шлюза по-умолчанию. Это тот узел сети, через который идет весь трафик по-умолчанию. Шлюз может быть изменен в результате взлома и переконфигурирования роутера, атак ICMP redirect или DHCP spoofing.
* GetHostName() - получение имени машины.
* GetPv4Address() - получение IP адреса машины.
* GetDnsList() - получение списка IP адресов DNS. Если были изменены адреса DNS серверов злоумышленник может перенаправить пользователя на поддельный сайт, который выглядит как точная копия легитимного и запросить у пользования ввести логин и пароль тут не спасет для шифрование посредством SSL.
* GetTracertTable() - получение таблицы маршрутизации. Нужно для дальнейшего анализа и выявления атак ARP spoofing и mac spoofing. При выполнение этих атак на атакуемой машине будет аномалия, а именно трафик вместо того чтобы сразу пойти на шлюз полмолчания будет проходить еще через один узел сети.
* GetArpTable() - получение ARP таблицы содержащий коллекцию Dto Host(рисунок 3). Нужно для дальнейшего анализа и выявления атак ARP spoofing и mac spoofing. При выполнение этих атак на атакуемой машине будет аномалия, а именно у 2 узлом с разными IP адресами будут одинаковые MAC адреса.
* GetHostInformation() - Получение информации об узле сети в виде Dto HostInformation(рисунок 4).

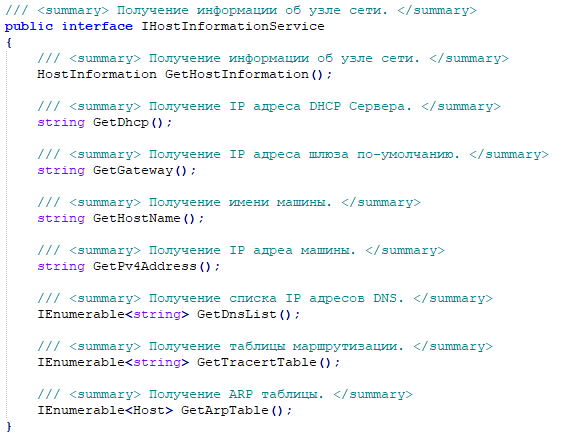


Рисунок 2 – Интерфейс IHostInformationService

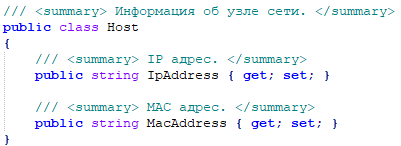


Рисунок 3 – Dto Host

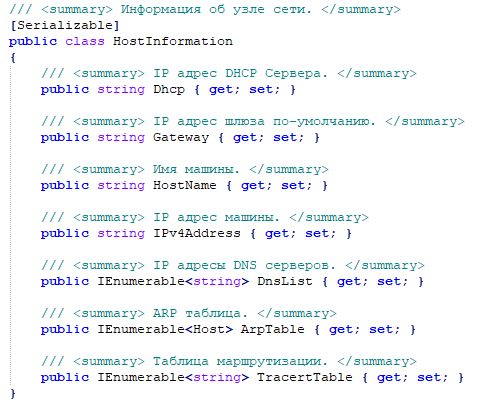


Рисунок 4 – Dto HostInformation

IWindowsManager – Интерес содержит в себе контракт для работы со специфическими вещями для ОС Windows который нет в стандартный библиотека .net такие как работа с протоколами ARP и ICMP (рисунок 5):

GetArpTable() - получение ARP таблицы для текущего узла сети, содержащий коллекцию Dto Host(рисунок 3)

GetTracertTable() - получение таблицы маршрутизации для текущего узла сети, до шлюза по умолчанию.

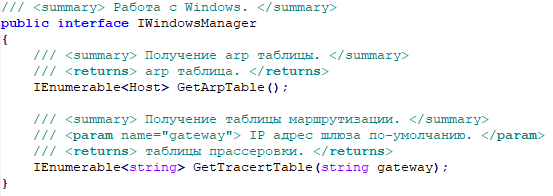


Рисунок 5 – Интерфейс IWindowsManager

IHttpClient – Интерес содержит в себе контракт для отправки, данный на сервер посредствам протокола HTTP (рисунок 6). Содержит в себе сигнатуру метода SendHostInformation для отправики Dto HostInformation с информацией о текущем узле сети на сервер для дальнейшей валидации.

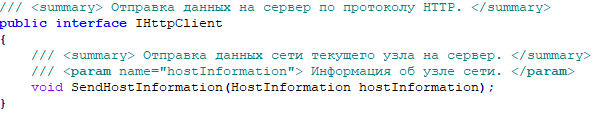


Рисунок 6 – Интерфейс IWindowsManager

## **Описание интерфейсов для Windows.**

WindowsHttpClientMock – класс реализующий интерфейс IHttpClient (рисунок 7), так как сервер еще не реализован, класс представляет собой заглушку для тестирования работоспособности сервиса. Класс WindowsHttpClientMock содержит реализацию метода SendHostInformation (рисунок 6), данный метод серилизует экземпляр класса HostInformation в формат json и сохраняет его в файл с имением текущей даты и времени.

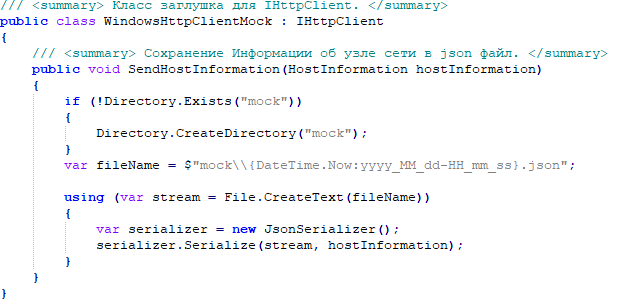


Рисунок 7 – Класс WindowsHttpClientMock

WindowsCmdManager – класс реализующий интерфейс IWindowsManager (рисунок 8). WindowsCmdManager – пример реализации интрефейса IWindowsManager с помощь стандартный утилит windows, а именно arp.exe и tracert.exe. В методах GetTracertTable и GetArpTable происходит вызов этих утилит с перенаправлением потока вывода на методы ParseTracert и ParseArp и дальнейшего парсинга результата работы утилит arp.exe и tracert.exe (рисунок 9).



Рисунок 8 – Класс WindowsCmdManager

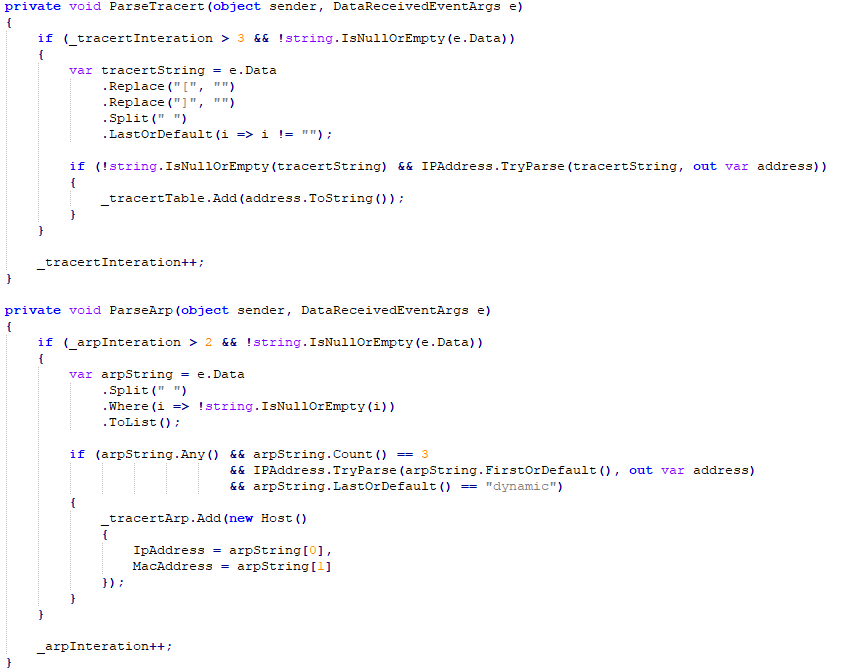


Рисунок 9 – Класс WindowsCmdManager

WindowsHostInformationService – класс реализующий интерфейс IHostInformationService (рисунок 10). Класс WindowsHostInformationService содержит в себе методы получающие информацию об узле сети. В классе WindowsHostInformationService есть две приватный переменный доступных только для чтения \_ipInterfaceProperties типа IPInterfaceProperties и \_manager типа IWindowsManager. В конструкторе с помощью такой техники как DI (Dependency injection) происходит внедрение зависимостей, делается это для большей гибкости и для удобства написания Unit тестов, так как можно указать любую реализацию реализующий интерфейс. Так же класс WindowsHostInformationService сдержит реализацию всех методов интерфейса IHostInformationService. Методы GetTracertTable и GetArpTable внутри себя вызывают методы из перемены \_manager, где может быть любой экземпляр класса реализующий интерфейс IWindowsManager.

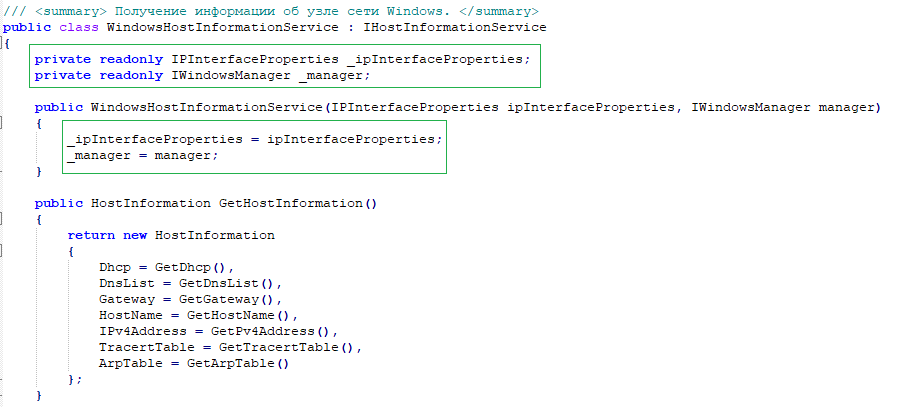


Рисунок 10 – Класс WindowsHostInformationService

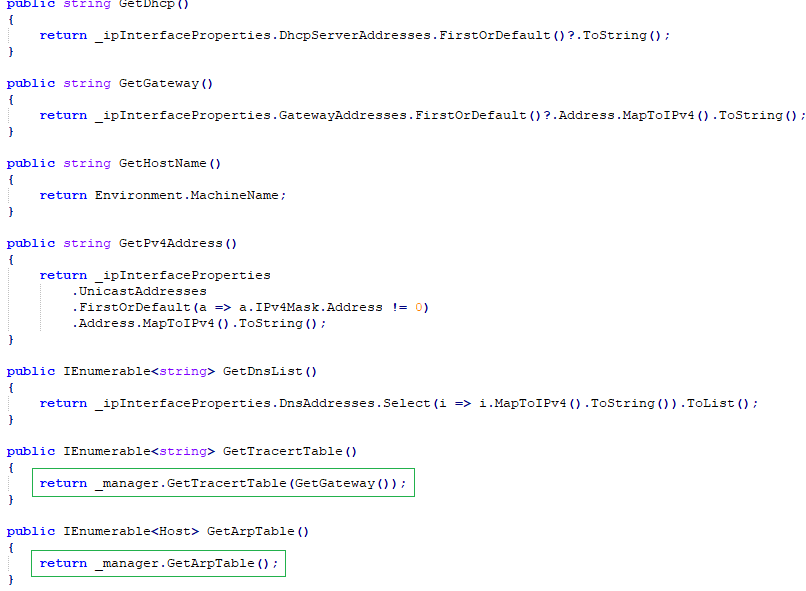


Рисунок 11 – Класс WindowsHostInformationService

## **Описание NetworkMonitor.WindowsService**

NetworkMonitor.WindowsService – сборка для создания клиента в виде службы window. Для этого необходимо добавить к проекту следующие Nuget пакеты Microsoft.Extensions.Hosting, Microsoft.Extensions.Hosting.WindowsServices и ссылка на проекты NetworkMonitor.Common и NetworkMonitor.Implementation. На рисунке 12 изображен кода из файла Program.cs в котором происходит конфигурация сервиса. Получение параметров из файла appsettings.json. И внедрение зависимостей в IoC контейнер. И также добавление WindowsWorker в котором будет выполнятся все логика. Рисунок 13. На рисунке 13 показан код работы метода ExecuteAsync в котором происходит получение настроек узла и отправку их на сервер (пока с помощью заглушки пишется в json файл). Затем идет задержка, значение которой берется из файла конфигурации и итерация повторяется. Для создания сервиса Windows необходимо скомпилировать сборку NetworkMonitor.WindowsService и полнить bat скрипт правами администратора указав в нем путь до NetworkMonitor.WindowsService.exe “create NetworkMonitor.WindowsService.bat” код которого представлен на рисунке 14. После чего можно запустить утилиту services.msc и убедиться, что создана новая служба, которую можно настроить и запускать или выключать. По умолчанию для служб windows используется папка C:\Windows\System32. Можно увидеть, что NetworkMonitor.WindowsService создал папку mock и сохраняет туда значение Dto HostInformation серилизуя их в json файл рисунок 16, рисунок 17. Запустив утилиту Event Viewer можно увидеть что системная служба пишет логи о своей работа рисунок 18.



Рисунок 12 – Конфигурация сервиса NetworkMonitor.WindowsService



Рисунок 13 – Конфигурация сервиса NetworkMonitor.WindowsService



Рисунок 14 – Скрипт create NetworkMonitor.WindowsService.bat

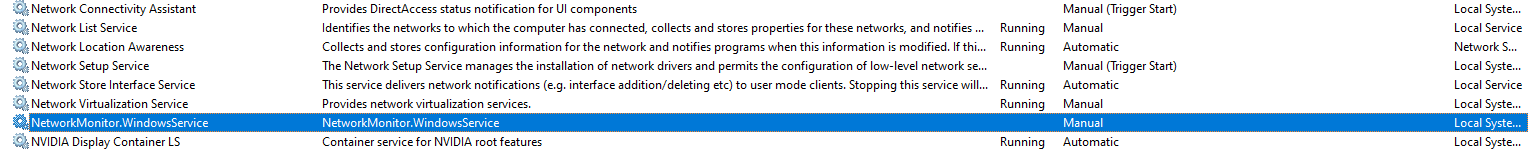


Рисунок 15 – Скрипт services.msc

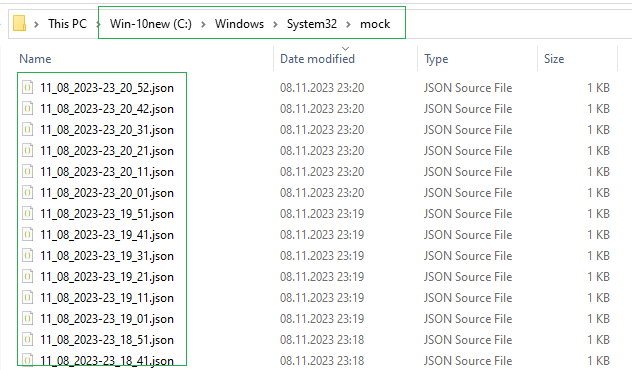


Рисунок 16 – Папка mock.



Рисунок 17 – Пример json файла.

**ЗАКЛЮЧЕНИЕ**

Основная идея разработки написать сетевое клиент-серверное приложение, которое позволяло бы централизованное выявлять атаки MITM (“человек посередине») и оперативно реагировать на инциденты. В статье описывается написание первой части этого приложения, а именно клиента, который планируется устанавливать на каждый узел сети, для отправки информации о сетевых настройках, а также ARP таблицы и таблицы трассировки. Описана архитектура приложения клиента, а также показа процесс компиляции и запуска клиента в качестве системной службы windows.

**Литература**

1. Mvah F. et al. GaTeBaSep: game theory-based security protocol against ARP spoofing attacks in software-defined networks //International Journal of Information Security. – 2023. – С. 1-15.
2. Galal A. A., Ghalwash A. Z., Nasr M. A New Approach for Detecting and Mitigating Address Resolution Protocol (ARP) Poisoning //International Journal of Advanced Computer Science and Applications. – 2022. – Т. 13. – №. 6. Maximov R.V., Sokolovsky S.P., Telenga A.P. Model of client-server information system functioning in the conditions of network reconnaissance. CEUR Workshop Proceeding. 2019. pp. 44-51.
3. Stepanov P.P. Attack on the Address Resolution Protocol / Stepanov P.P., Nikonova G.V., Pavlychenko T.S., Gil A.S. // 2020 International Conference Engineering and Telecommunication (En&T), 2020, pp. 1-3.
4. Stepanov P.P. The problem of security address resolution protocol / P.P. Stepanov, G.V. Nikonova, T.S. Pavlychenko, A.S. Gil // Journal of Physics: Conference Series. - 2021, Vol. 1791, p.p. 1-8.
5. Степанов, П. П. Особенности работы протокола разрешения адресов в компьютерных сетях / П. П. Степанов, Г. В. Никонова, Т. С. Павлюченко, В. В. Соловьев // Программная инженерия. – 2022. Том 13, № 5. – С. 211-218.
6. Shah Z, Cosgrove S. Mitigating ARP Cache Poisoning Attack in Software-Defined Networking (SDN): A Survey. *Electronics*. 2019; 8(10):1095. https://doi.org/10.3390/electronics8101095.
7. Биджиева С.Х., Шебзухова К.В. Сетевые протоколы передачи данных: преимущества и недостатки // Тенденции развития науки и образования. 2022. Т. 86. № 1. С. 43–45. doi: 10.18411/trnio-06-2022-14.
8. Hijazi, S.; Obaidat, M. Address resolution protocol spoofing attacks and security approaches: A survey. Secur. Priv. 2019, 2, e49
9. Дементьев В.Е., Чулков А.А. Кибервоздействия на протоколы сетей передачи данных // Изв. ТулГУ. Технические науки. 2020. № 10. С. 245–254.
10. Барабошкин Д.А., Бакаева О.А. Анализ алгоритмов шифрования данных // За нами будущее: взгляд молодых ученых на инновационное развитие общества: сб. ст. науч. конф. 2022. Т. 2. С. 449–452.
11. Снейдер, И. Эффективное программирование TCP/IP. Библиотека программиста : пер. с анг. – М.: ДМК Пресс. – 2019. – 322 с.
12. Барабошкин Д.А., Бакаева О.А. Разработка комбинированного алгоритма шифрования мультимедийных данных в процессе их передачи // Математическое моделирование, численные методы и комплексы программ: сб. тр. X Междунар. науч. молодежн. школы-семинара им. Е.В. Воскресенского. 2022. С. 27–31. URL: https://conf. svmo.ru/files/2022/papers/paper05.pdf (дата обращения: 28.02.2023).
13. [Mujahid Shah](https://ieeexplore.ieee.org/author/37086803375), [Sheeraz Ahmed](https://ieeexplore.ieee.org/author/37086133637), [Khalid Saeed](https://ieeexplore.ieee.org/author/37086692637), [Muhammad Junaid](https://ieeexplore.ieee.org/author/37086803611), [Hamayun Khan](https://ieeexplore.ieee.org/author/37086803793), [Ata-ur Rehman](https://ieeexplore.ieee.org/author/37086804526). Penetration Testing Active Reconnaissance Phase – Optimized Port Scanning With Nmap Tool.  [// 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)](https://ieeexplore.ieee.org/xpl/conhome/8671662/proceeding), 2019. Sukkur, Pakistan. DOI: [10.1109/ICOMET.2019.8673520](https://doi.org/10.1109/ICOMET.2019.8673520).

1. Степанов Петр Петрович, старший преподаватель, федерального государственного автономного учреждения высшего образования «Омский государственный технический университет» (ОмГТУ), г. Омск, Россия, E-mail: omsk.petr@gmail.com [↑](#footnote-ref-1)
2. Никонова Галина Владимировна, кандидат технических наук, доцент, федерального государственного автономного учреждения высшего образования «Омский государственный технический университет» (ОмГТУ), г. Омск, Россия. E-mail: nikonova@omgtu.ru [↑](#footnote-ref-2)