УДК 004.05 **DOI: 10.18413/2518-1092-2025-10-2-0-5**

Головинский С.А.
Маслова М.А.
Лагуткина Т.В.

ИСПОЛЬЗОВАНИЕ ХЕШ-ТАБЛИЦ В МЕХАНИЗМЕ ЗАШИТЫ ОТ DOS-АТАК НА ПРИМЕРЕ ЯЗЫКА PYTHON

Севастопольский государственный университет,
ул. Университетская, 33, г. Севастополь, 299053, Россия

e-mail: tositrent164@gmail.com, mashechka-81@mail.ru, t.v.lagutkina@mail.sevsu.ru

Аннотация
В данной статье рассматривается актуальная проблема безопасности хеш-таблиц при DoS-атаках, вызванных преднамеренными коллизиями. Хеш-коллизии могут использоваться злоумышленниками для значительного замедления работы системы, так как при этом увеличивается время поиска и вставки данных в хеш-таблицах. Основное внимание уделено механизмам защиты, применяемым в Python, где для предотвращения подобных атак используется рандомизированная соль в хеш-функциях. В работе детально анализируется принцип работы хеш-таблиц, природа возникновения коллизий и особенности реализации рандомизированнного хеширования. Объясняются основные концепции и приводятся практические примеры на Python. Исследуются различные методы атак через коллизии и способы их предотвращения. Заключительная часть статьи содержит конкретные рекомендации по безопасному использованию хеш-таблиц и повышению общей безопасности информационных систем. Рассматриваются как технические аспекты реализации защитных механизмов, так и организационные меры по защите от Dos-атак. Статья будет полезна разработчикам программного обеспечения, специалистам по информационной безопасности и исследователям в области компьютерных наук.
Ключевые слова: информационная безопасность; безопасность; хеш-таблица; DoS-атака; коллизия; рандомизация хеша; Python; защита от DoS; хеш-функция
Для цитирования: Головинский С.А., Маслова М.А., Лагуткина Т.В. Использование хеш-таблиц в механизме зашиты от DoS-атак на примере языка Python // Научный результат. Информационные технологии. – Т. 10, №2, 2025. – С. 49-56. DOI: 10.18413/2518-1092-2025-10-2-0-5

Golovinskiy S.A.
Maslova M.A.
Lagutkina T.V.

USAGE OF HASH TABLES IN THE MECHANISM OF PROTECTION AGAINST DOS ATTACKS USING THE PYTHON LANGUAGE

Sevastopol State University,
33 Universitetskaya St., Sevastopol, 299053, Russia

e-mail: tositrent164@gmail.com, mashechka-81@mail.ru, t.v.lagutkina@mail.sevsu.ru

Abstract
This article addresses the current problem of hash table security in the context of DoS attacks caused by deliberate collisions. Hash collisions can be exploited by attackers to significantly slow down system performance, as this increases the time for searching and inserting data in hash tables. The main focus is on protective mechanisms implemented in Python, where randomized salts are used in hash functions to prevent such attacks. The paper provides a detailed analysis of how hash tables work, the nature of collision occurrences, and the features of implementing randomized hashing. Key concepts are explained, and practical examples in Python are provided. Various collision attack methods and their prevention strategies are explored. The concluding part of the article contains specific recommendations for the secure use of hash tables and enhancing the overall security of information systems. Both the technical aspects of implementing protective mechanisms and the organizational measures for protection against DoS attacks are considered.

The article will be useful for software developers, information security specialists, and researchers in the field of computer science.
Keywords: information security; security; hash table; Ddos attack; collision; hash randomization; Python; DoS protection; hash function
For citation: Golovinskiy S.A., Maslova M.A., Lagutkina T.V. Usage of hash tables in the mechanism of protection against dos attacks using the Python language // Research result. Information technologies. – T. 10, №2, 2025. – P. 49-56. DOI: 10.18413/2518-1092-2025-10-2-0-5

ВВЕДЕНИЕ

Поддержание связи в Интернете основано на передаче данных. К сожалению, это также означает, что большой объем информации широко доступен для трансфертов между устройствами и серверами, и, следовательно, подвержен возможности атак. На данном этапе защита информации является критически необходимой из-за ряда видов атак, конфиденциальности пользователей и предотвращения доступа для лиц, не имеющих на это права.

Для обеспечения безопасности передачи каких-либо данных существуют различные методы, которые помогают защищать информацию от несанкционированного доступа, возможных утечек, краж и подделки. К основным из них относиться:

– шифрование, является одним из самых эффективных способов защиты данных при передаче по сети, хранения на устройствах [1];

– межсетевые экраны или так называемые файрволы, они служат фильтром для исходящего и входящего трафика, при этом блокируют запрещенные и подозрительные соединения с предотвращением несанкционированного доступа к сети;

– DLP или система предотвращения утечек. Она анализирует потоки информации и блокирует передачу какой-либо конфиденциальной информации и данных за ее пределы;

– электронная подпись и цифровые сертификаты. Их используют для узнаваемости участников обмена и предотвращения использования чужих данных;

– протоколы безопасности. Их используют для шифрования веб-трафика между серверами и браузерами (например HHTTPS), создание шифровального канала между точками обмена, при этом защищая данные даже в общественных местах – VPN, создание защищенного соединения между сервером и клиентом (SSL/TLS);

– многофакторная аутентификация. С помощью нее формируется добавление дополнительных уровней проверки пользователей, что положительно влияет на уменьшение риска взлома аккаунтов пользователей. Проводится с помощью биометрии или SMS-кода на телефон пользователя;

– хеширование позволяет проверять исходность начальных данных, не были ли эти данные изменены и позволяет защитить от кражи с помощью преобразования данных в уникальный хеш-код, который невозможно восстановить в исходный вид. Данный метод является одним из основных в обеспечении безопасности передачи данных. С помощью него хранят пароли и методы контроля на целостность информации.

ОСНОВНАЯ ЧАСТЬ

Атака отказа в обслуживании или DOS-атака является одной из тех, с помощью которой можно обойти шифрование несмотря на разновидность шифров, защищенности алгоритмов хеширования. Злоумышленниками используются слабые места системы и создаются искусственные коллизии и более замедленная работа хеш-таблиц, что влияет на возможность создания угроз для безопасности данных и производительности серверов. Т.е. с помощью данной атаки происходит большая нагрузка, которая не дает возможность обработать запросы от пользователей [2, 3].

Почему же так часто используют DOS-атаки? Они являются доступными в финансовой составляющей, поэтому их часто используют после проведения конкурентной разведки для

ликвидации конкурентов и других незаконных действий. Для понимания работы данной атаки необходимо понимать работу ее структуры данных. Есть хеш-таблица, которая является структурой данных. В ней хранятся элементы в виде пары ключ – значение. Где ключ – это неповторяющееся число, использующееся для индексации значений. В свою очередь значение – это данные, связанные с данным ключом [4]. Хэш таблица основана на бакетах, а точнее динамических массивах – структуры данных, под которые каждый язык программирования выделяет свой участок памяти, также при добавлении элементов в данный массив происходит увеличение затрат по памяти. На рисунке 1 представлена концептуальная схема работы хэш-таблицы.

Концептуальная сложность алгоритма добавления элемента в таблицу равняется «O(1)» [5]. Данную нотацию «O(1)» можно прочитать как сложность порядка 1, или алгоритм выполняется за постоянное/константное время. Поиск элемента в хэш-таблице по ключу в лучшем случае выполняется за «O(1)». Данные операции так эффективны благодаря алгоритмам хэширования. Когда необходимо добавить новое значение - сначала вычисляется хэш-функция от ключа. Ключ с помощью хеш-функции преобразуется в числовое значение, проводит математические вычисления в виде вычисления результата остатка от деления хеш-значения на его длину динамического массива, получая результат в виде индекса, в который и вставляются значения. Далее переходим ко второму динамическому массиву. В него помещается значение с вычисленного индекса. Размер динамического массива можно менять, если необходимо хранить множество значений по данному ключу. При этом могут возникнуть коллизии:

– индексные коллизии – это ситуации, когда при разных ключах они дают одинаковый хеш-индекс из-за того, что количество возможных ключей больше, чем размер самой таблицы. Для устранения данного недостатка используется открытая адресация или цепочка. Это дает возможность устранить коллизии, но при этом увеличивается сложность выполняемых операций до самой плохой O(n), где n – это количество элементов в ячейке.

– коллизии значений – когда в одну ячейку таблицы (с одним индексом) помещаются несколько значений, принадлежащих разным ключам. Для хранения нескольких значений по одному индексу используются структуры данных, такие как списки или деревья.

Хэш-таблицы является распространенным инструментом для выполнения DoS атак на разные сервисы обработки данных. Одним из методов организации атаки является целенаправленное заполнение хеш-таблицы данными, которые вызывают множественные коллизии, то есть распределяются в один слот. В результате снижается производительность хеш-таблицы: время поиска элемента возрастает до O(n), а вставка нового элемента с коллизией требует времени $O(n^2)$. Потребление памяти также увеличивается из-за так называемого «заливания хешей» (hash-flooding). Для проведения такой атаки злоумышленник должен иметь возможность добавлять произвольные данные в хеш-таблицу и находить коллизии хеш-функции [6-8].

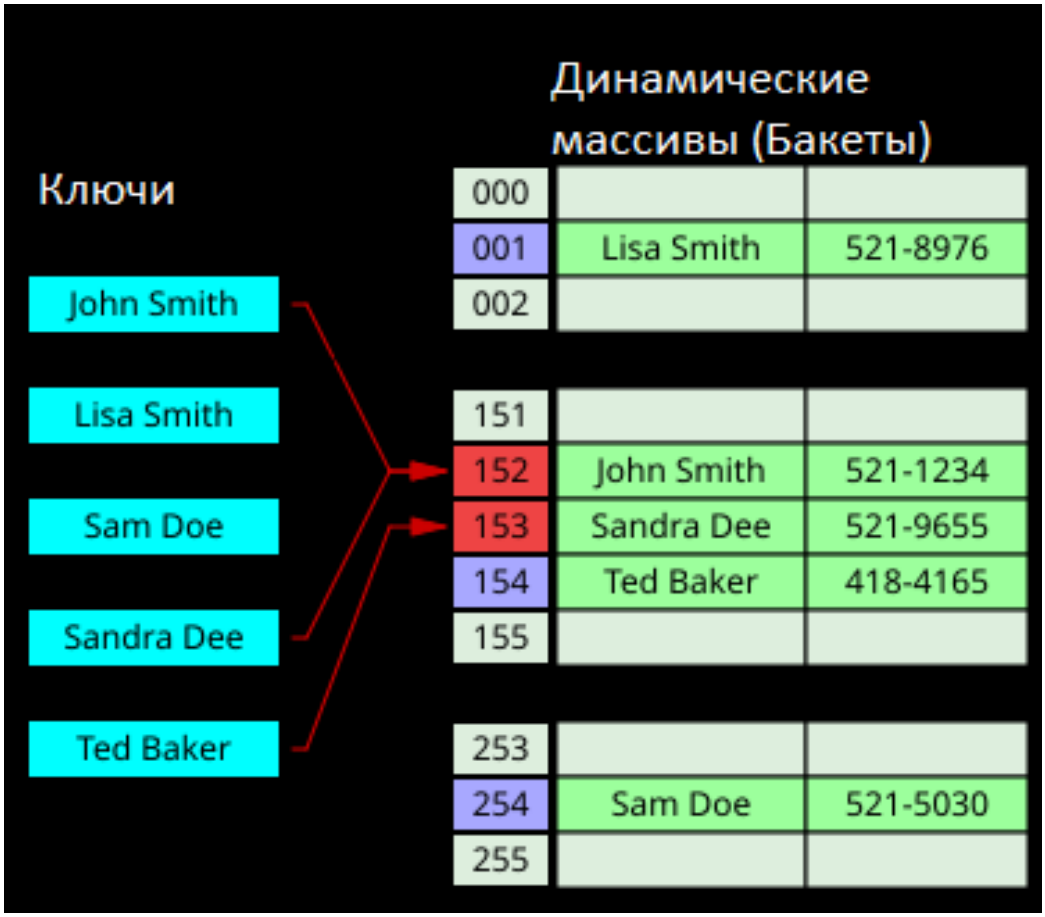

Рис. 1. Схема работы хэш-таблицы
Fig. 1. Hash table operation diagram

Данную атаку называют hash DoS – атака или атака на алгоритмическую сложность (algorithmic complexity attack). А разных источниках отслеживается мнение по поводу данной атаки на предмет того, что она не является атакой, которую должна выполнять хеш-функция. Для этого можно использовать различные методы, такие как структуры данных с меньшей сложностью. Не так действенно, но имеет место быть. Приведем пример. Имеется сбалансированное дерево – АВЛ-дерево или красно-чёрное дерево – гарантирует в любом случае O(log n). В данном случае важной необходимостью есть учет новых атак, например, каскадный ребаланс дерева с специально сформированными данными, но даже в этом случае получится лучше результат – O(n log n),

Рассмотрим некоторые существующие способы создания коллизий:

– метод полного перебора, когда известна хеш-функция и ее длина (небольшая). Сюда так же включается подход meet-in-the-middle, для возможности вычислять подходящие коллизии;

– нахождение математическим путем, когда хеш-функция не устойчива к коллизиям.

Некриптографические хеш-функции в принципе относительно просты, и для нахождения коллизий бывает достаточно обратить алгоритм, выполнив его в обратном порядке. На рисунке 2 представлена топологическая схема Dos-атаки с заранее подготовленной структуры данных с коллизиями.

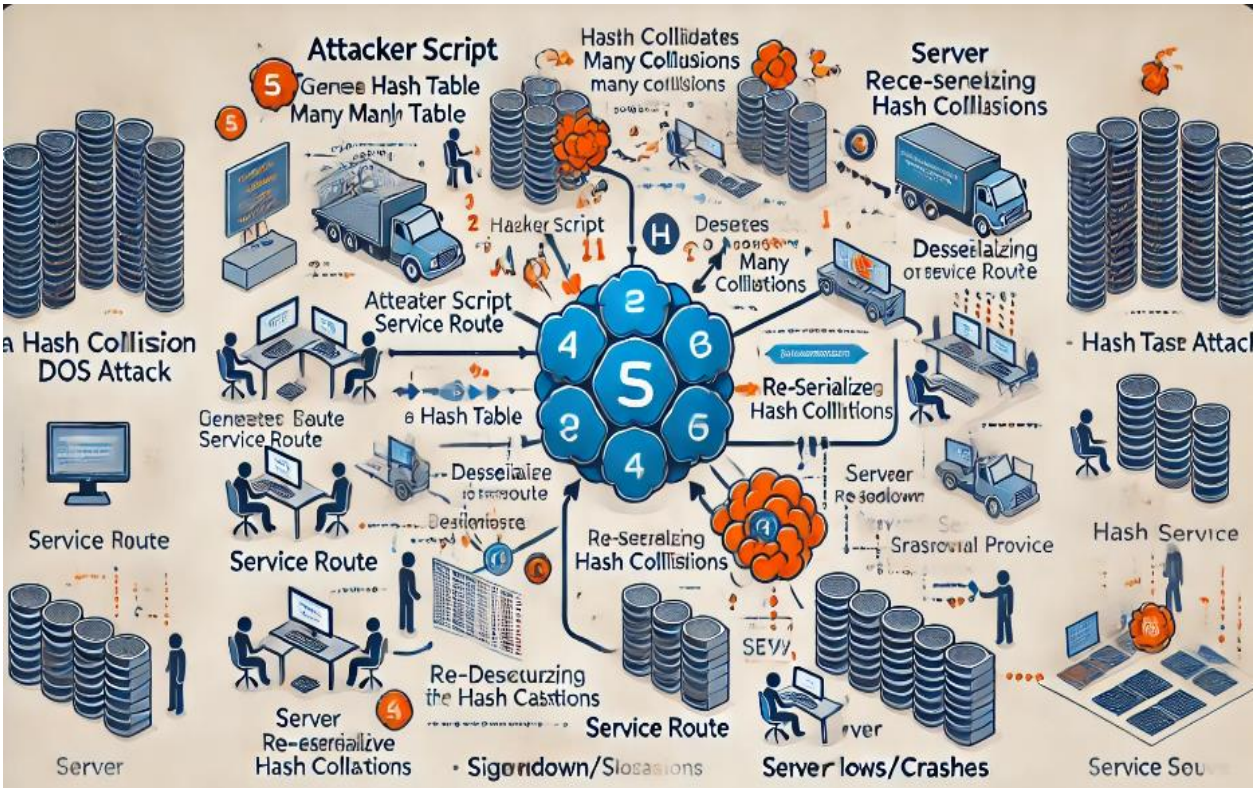

Рис. 2. Схема Dos атаки от злоумышленника до сервиса
Fig. 2. Dos attack diagram from the attacker to the service

Злоумышленник умышленно запускает скрипт, при помощи которого создаются структуры данных с коллизиями. Результат скрипта отправляется по открытому маршруту на сервис, обрабатывающий данные. Сервис сериализует поврежденные данные, вследствие чего происходит засорение памяти на сервере. Аллокатор памяти не справляется с той нагрузкой, которая пришла ему на вход, в результате чего сервер тормозит. В результате происходят ошибка переполнение стэка – stack overflow, концептуально возможно также переполнение в куче – структуре данных, которой управляет аллокатор памяти [9-11]. Для добавления объектов в кучу с помощью аллокатора памяти с адресами представлена на рисунке 3.

В языке программирования Python существует встроенный механизм, предназначенный для защиты от коллизей в структурах данных. Данный язык программирования использует механизм так называемый – рандомизация хешей. Он добавляет случайную соль (salt) к хеш-функциям для строковых ключей [12]. При использовании данной рамдомной salt, обеспечивающая различное поведение для каждой сессии, а также дающая возможность предотвращать предсказуемое создание коллизий. Когда происходит запуск интерпретатора языка Python, то для каждой новой сессии происходи генерация случайной соли. Она в свою очередь добавляется к хеш-значению строк [13]. Это дает возможность создавать предсказуемые коллизии за счет соли. Так как строки, которые до этого могли вызывать коллизии в хеш-таблице, теперь могут получать различные хеш-значения при каждом запуске Python. Данный механизм является встроенным в стандартные хеш-таблицы Python, которые так же включают и словари, и множества. Поэтому пользователю не приходится включать его отдельно. Когда злоумышленник пытается в Python использовать словари или множества, он не может просто создать данные, которые приведут к одинаковым хеш-значения из-за того, что salt меняется при каждом новом запуске. Следовательно это делает язык программирования Python устойчивым к DOS – атакам с использованием преднамеренных коллизий.

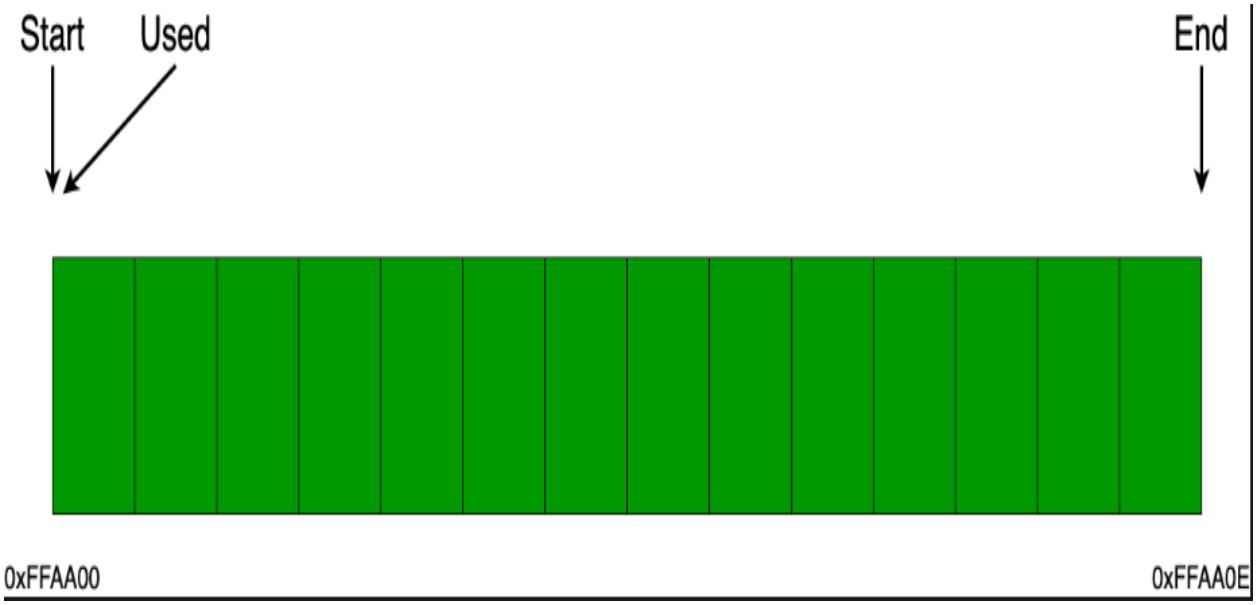

Рис. 3. Незагруженный аллокатор памяти
Fig.3. Unloaded memory allocator

Данные алгоритмические DOS – атаки основываются на идее специального создавания коллизий в хеш-таблицах. Как уже упоминалось, с помощью них можно замедлить обработку данных и увеличить потребление на сервере ресурсов, что приведет к увеличению времени выполнения всех операций и, следовательно, значительному увеличению нагрузки на память. Такой способ использования рандомизации для защиты от атак на хеш-таблицы используется так же и другими языками программирования и феймворками. Например, это Java и Ruby. Особенно они используются в их веб – приложениях для того, чтобы справится с атаками на алгоритмическую сложность.

Например, в языке Java хеш-коллизии решаются через хеш-таблицы и бинарные деревья для хранения конфликтующих значений. Python, в свою очередь, использует рандомизацию для предотвращения создания предсказуемых коллизий. Защита с использованием соли работает эффективно в локальных приложениях и на серверной стороне, где данные ограничены одной сессией или перезапуском сервера. Тем не менее, в распределенных системах, где рандомизация может быть зафиксирована для унитарности, такой подход может быть нерентабельным, что делает рискованным анализ атак на коллизии в зависимости от типа архитектуры системы. Кроме того, для обеспечения максимальной безопасности необходимо применять ограничение размера хеш-таблиц, защиту от перегрузки по числу элементов и другие способы мониторинга и управления запросами на стороне веб-сервера или приложения. Система станет значительно более защищенной при использовании библиотек и фреймворков, учитывающих безопасность хеширования и способные обеспечивать защиту от алгоритмических атак [14, 15].

ЗАКЛЮЧЕНИЕ

В заключении к вышеперечисленному, были рассмотрены принципы безопасности алгоритмов хэширования для защиты от DoS атак. В современных условиях увеличивающегося числа кибератак на реализующие алгоритмическую сложность уязвимости целевые системы становится важным понимание значимости защиты хеш-таблиц, отчасти являющихся неотъемлемой частью практически всех современных языков программирования и целевых платформ. Хеш-таблицы необходимы для создания структур, обеспечивающих быстрый доступ к определенным данным, но предсказуемость хеш-функций делает их уязвимыми к ряду atack pattern, таких как hash flooding, замедляющей обработку и приводящих к DoS. По этой причине

использовалась случайная соль для защиты в языке программирования Python. Она не требует дополнительных действий со стороны пользователя для включения, а добавляет автоматическое внедрение случайной величины, изменяющей результат хеширования при каждом перезапуске интерпретатора. Для того что бы сделать систему более устойчивой к DOS – атакам и менее предсказуемой для хакеров и злоумышленников, необходимо использовать рандомизацию уровней хешей. Они усложняют манипулирование параметрами хеша для создания известных криптографических коллизий. Очень хорошо использовать автоматически для хеш функций случайную соль. Так как дает безопасность по умолчанию, защиту для стандартных библиотек Python от множества классов атак, при этом не требует дополнительных конфигураций или внешних библиотек.

Таким образом с помощью данного механизма получается комплексная локальная проверка, средство обхода для Python – приложений. Дает возможность повысить безопасность для самых популярных программных структур, используемых в разных странах мира. С помощью рандомизации хешей в Python, которая является важнейшим элементом для защиты данных дает возможность хорошей, надежной и простой защиты созданной командой разработчиков Python. В целом, эта технология является страстным примером того, как автоматизация и стандартизация защиты могут бороться с угрозами на самом высоком уровне.

Список литературы

1. Кузьминых, Е.С. Анализ непробиваемых алгоритмов шифрования / Е.С. Кузьминых, С.П. Ильина, М.А. Маслова // Научный результат. Информационные технологии. – 2024. – Т. 9, № 1. – С. 10-18. DOI: 10.18413/2518-1092-2024-9-1-0-2
2. DoS-атака [Электронный ресурс]. URL: https://stormwall.pro/rsources/terms/attacks/dos
3. Надейкина, В.С. Анализ систем обнаружения и предотвращения вторжения с открытым кодом для интеграции с отечественными операционными системами / В.С. Надейкина, М.А. Маслова // Научный результат. Информационные технологии. – 2024. – Т. 9, № 2. – С. 41-48. DOI: 10.18413/2518-1092-2024-9-2-0-5
4. Хэш таблицы [Электронный ресурс]. URL: https://codechick.io/tutorials/dsa/dsa-hash-table
5. Описание сложности алгоритмов [Электронный ресурс]. URL: https://habr.com/ru/articles/444594
6. Некриптографические хеш-функции и DoS атака на них [Электронный ресурс]. URL: https://habr.com/ru/articles/178955
7. Коллизионная атака [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B0%D1%82%D0%B0%D0%BA%D0%B0
8. Модель памяти в языках программирования [Электронный ресурс]. URL: https://tproger.ru/articles/memory-model
9. Работа памяти в Python [Электронный ресурс]. URL: https://habr.com/ru/articles/721804/
10. Переполнение Стэка [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D1%82%D0%B5%D0%BA%D0%B0
11. Стэк и куча – в чем разница [Электронный ресурс]. URL: https://wiki.merionet.ru/articles/stek-i-kuca-v-cem-raznica
12. Солим пароли [Электронный ресурс]. URL: https://habr.com/ru/articles/145648/
13. Язык программирования Python. Часть 2. Компиляторы и Интерпретаторы [Электронный ресурс]. URL: https://verity.by/news/yazyk-programmirovaniya-python-chast-2-kompilyatory-i-interpretatory/
14. Детерминанты развития экономики России в условиях цифровой трансформации и обеспечения технологического суверенитета / Е.Н. Макаренко, И.А. Полякова, И.А. Кислая [и др.]. – Ростов-на-Дону: Ростовский государственный экономический университет "РИНХ", 2023. – 546 с.
15. Маслова, М.А. Проблемы облачных сервисов и методы защиты от рисков и угроз / М.А. Маслова, Е.С. Кузьминых // Научный результат. Информационные технологии. – 2022. – Т. 7, № 3. – С. 14-22. – DOI 10.18413/2518-1092-2022-7-3-0-2.

References

1. Kuzminykh, E.S. Analysis of impenetrable encryption algorithms / E.S. Kuzminykh, S.P. Ilina, M.A. Maslova // Research result. Information technologies. – T. 9, №1, 2024. – P. 10-18. DOI: 10.18413/2518-1092-2024-9-1-0-2
2. Dos-Attack [Electronic resource]. URL: https://stormwall.pro/rsources/terms/attacks/dos
3. Nadejkina, V. S. Analysis of open-source intrusion detection and prevention systems for integration with russian operating systems / V. S. Nadejkina, M. A. Maslova // Research result. Information technologies. – T.9, №2, 2024. – P. 41-48. DOI: 10.18413/2518-1092-2024-9-2-0-5
4. Hash maps [Electronic resource]. URL: https://codechick.io/tutorials/dsa/dsa-hash-table
5. Description of the complexity of the algorithms [Electronic resource]. URL: https://habr.com/ru/articles/444594
6. Non-cryptographic hash functions and DoS attack on them [Electronic resource]. URL: https://protect.htmlweb.ru/des.htm
7. Collision attack [Electronic resource]. URL: https://habr.com/ru/articles/178955
8. Memory model in programming languages [Electronic resource]. URL: https://tproger.ru/articles/memory-model
9. Memory operation in Python [Electronic resource]. URL: https://habr.com/ru/articles/721804/
10. Stack overflow [Electronic resource]. URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81%D1%82%D0%B5%D0%BA%D0%B0
11. What's the difference between stack and heap Electronic resource]. URL: https://wiki.merionet.ru/articles/stek-i-kuca-v-cem-raznica
12. Password salts (Advanced Encryption Standard) works [Electronic resource]. URL: https://habr.com/ru/articles/145648/
13. Python programming language. Part 2. Compilers and Interpreters [Electronic resource]. URL: https://verity.by/news/yazyk-programmirovaniya-python-chast-2-kompilyatory-i-interpretatory/
14. Determinants of the development of the Russian economy in the conditions of digital transformation and ensuring technological sovereignty / E.N. Makarenko, I.A. Polyakova, I.A. Kislaya [and others]. – Rostov-on-Don: Rostov State Economic University "RINH", 2023. – 546 p. – ISBN 978-5-7972-3086-1. – EDN SHIVS.
15. Maslova, M.A. Problems of cloud services and methods of protection against risks and threats / M.A. Maslova, E.S. Kuz'minyh // Research result. Information technologies. – T.7, №3, 2022. – P. 14-22. DOI: 10.18413/2518-1092-2022-7-3-0-2

Головинский Семён Антонович, студент первого курса магистратуры кафедры «Информационная безопасность», Севастопольский государственный университет, г. Севастополь, Россия
Маслова Мария Александровна, доцент кафедры «Информационная безопасность», Севастопольский государственный университет, г. Севастополь, Россия
Лагуткина Татьяна Владимировна, старший преподаватель кафедры «Информационная безопасность», Севастопольский государственный университет, г. Севастополь, Россия

Golovinsky Semyon Antonovich, first-year Master's Student of the Department of Information Security, Sevastopol State University, Sevastopol, Russia
Maslova Maria Aleksandrovna, Associate Professor of the Department of Information Security, Sevastopol State University, Sevastopol, Russia
Lagutkina Tatiana Vladimirovna, Senior Lecturer of the Department of Information Security, Sevastopol State University, Sevastopol, Russia