ИСПОЛЬЗОВАНИЯ УЯЗВИМОСТЕЙ ПРОТОКОЛА WEBAUTHN ДЛЯ ПОЛУЧЕНИЯ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

Панченко А. Р.[1]

DOI: 10.21681/2311-3456-2025-6-48-57

Цель статьи: *исследование текущего стандарта беспарольной аутентификации FIDO2 на наличие уязвимостей и подтверждение отсутствия у него доказательной устойчивости.*

Методы исследования: *анализ текущего стандарта беспарольной аутентификации на наличие уязвимостей. Реализация и эксплуатация обнаруженной уязвимости. Данная уязвимость базируется на использование модифицированной атаки «человек посередине» с использование вредоносного программного обеспечения и социальной инженерии.*

Результаты исследования: *проведен анализ текущего стандарта беспарольной аутентификации. В рамках данного анализа была сформирована формальная схема работы протокола WebAuthn. Была обнаружена уязвимость, позволяющая получить доступ к аккаунту легитимного пользователя, защищенного текущим стандартом FIDO2. Для реализации данной уязвимости использовалась модифицированная атака «человек посередине», в рамках которой был реализован сервер, занимающийся пересылкой сообщений во время проведения атаки. Было реализовано вредоносное программное обеспечение типа троянский конь. Данное вредоносное программное обеспечение предварительно было помещено в систему жертвы, где оно выдавало себя за браузер жертвы, с которым связался легитимный сервер. На стороне злоумышленника была реализована программа, считывающая запрос аутентификации в системе атакующего. Данная программа используется для переопределения функции во время процесса аутентификации и последующей пересылки запроса на сервер. В рамках реализации данной уязвимости и ее последующего выполнения было подтверждено отсутствие доказательной устойчивости у текущего стандарта беспарольной аутентификации FIDO2. Был предложен способ потенциальной защиты от данной уязвимости. Так же был предложен способ модификации сбора данных на этапе регистрации, для последующего уведомления об потенциальной атаке жертвы во время процедуры подписания.*

Научная новизна: *обнаруженная уязвимость подтверждает отсутствие доказательной устойчивости у текущего стандарта беспарольной аутентификации FIDO2. На данный момент все браузеры используют WebAuthn это означает, что данной уязвимости подвержены все современные браузеры. Предложен потенциальный способ защиты от данной уязвимости и способ предупреждения жертвы в рамках уже проводимой атаки.*

Ключевые слова: *беспарольная аутентификация, ассиметричное шифрование, Self-XSS, Python, FIDO2, CTAP2.1, Authorization Gesture.*

Введение

В современном мире все люди пользуются паролями. Все исследования последнего времени демонстрируют, что пароли устарели уже давно. «В июне 2024 года «Лаборатория Касперского» проанализировала 193 млн паролей, обнаруженных в публичном доступе на даркнет-ресурсах. Результаты исследования показали, что: почти половину из них (45 %, или 87 млн) мошенники смогут подобрать менее чем за минуту; большинство проанализированных паролей могут быть легко скомпрометированы с помощью умных алгоритмов; только 23 % (44 млн) комбинаций оказались достаточно стойкими: на их взлом ушло бы больше года» [1, с. 3]. Подсчет энтропии для паролей разной длины показывает, что 8 символов даже при самой большой мощности алфавита имеет значение энтропии 47,6 [1]. «Если вес пароля от 34 до 67, то пароль относится к категории «Хороший», а если более 67, то пароль считается надежным.» [1, с. 49]. Так же ситуацию осложняет человеческий фактор, связанный тем, что люди не стремятся делать сложные пароли самостоятельно, так как они слишком сложны для запоминания, хоть и более безопасные. «Так, по информации securitylab в 2008 году 84 % случаев взломов систем безопасности компаний и персональных страниц причиной была именно слабая парольная защита»[2]. Данный тезис подтверждается в статье [3, с. 79]: «Пароли, самостоятельно устанавливаемые пользователями, бывают качественными крайне редко. Сложность паролей, генерируемых различными информационными системами, на практике часто оказывается обесцененной самими работниками предприятия, записывающими

1 Панченко Александр Романович, аспирант, Южный Федеральный Университет «ЮФУ», Институт компьютерных технологий и информационной безопасности, г. Таганрог, Россия. ORCID: https://orcid.org/0009-0001-4720-5164. E-mail: alpanchenko@sfedu.ru

2 Снегуров А. В, Чакрян В. Х. Анализ устойчивости ко взлому современных механизмов парольной защиты операционных систем // Восточно-Европейский журнал передовых технологий. 2011 – No 10. – С. 27–29.

и сохраняющими выданные им пароли». Иногда для лучшего запоминания сложного пароля люди используют «информативные последовательности символов»[3], что тоже не самый надежный способ защиты данных. Важным моментом является тот факт, что небольшая модификация семантически значимой последовательности символов может превратить её в достаточно стойкий или даже очень стойкий пароль. Пример такого превращения фразы в пароль показан в статье [4, с. 159]. Суровая реальность показывает, что данным методом люди зачастую пренебрегают. При этом число сервисов, учетных записей и аккаунтов с каждым годом неуклонно растет, и для каждого их них требуется свой, в идеальном варианте, уникальный пароль. Но практика показывает, что часто один пароль используется во множестве сервисов и аккаунтов либо с минимальными изменениями, либо вообще без них. «Мы обнаружили, что повторное использование и модификация паролей являются очень распространенным явлением (наблюдается у 52 % пользователей)»[4]. «Также очень важно не использовать одинаковые пароли для разных учётных записей, ведь если один пароль будет скомпрометирован, это может повлечь за собой взлом и других учётных записей, где соответствующий пароль используется» [5]. А если учитывая факт про сложность паролей, то ситуация становится критической [6, с. 66]. Статистика фишинга подтверждает всю критичность ситуации на данный момент: «Что касается глобальных тенденций, то в 2022 году APWG зафиксировала около 4,7 миллиона фишинговых атак. С 2019 года наблюдается ежегодное увеличение числа атак более чем на 150 %» [7, с. 3]. Важно упомянуть еще и технические аспекты, связанные с ограничениями при создании пароля в разных ОС. Численные характеристики данного аспекта показаны в статье [8]. Проанализировав все минусы парольной аутентификации можно прийти к выводу, что на данный момент направление, связанное с беспарольной аутентификацией [9, 10, с. 144] – переходом от фактора знания (пароля), к фактору владения является наиболее перспективным в сложившихся условиях.

Постановка задачи

В настоящей работе рассматривается схема и принцип работы стандарта FIDO2. Данные об текущем стандарте были получены из документации стандарта с официального сайта FIDO, а также из статьи [11]. Информация о протоколе WebAuthn была получена из официального сайта консорциума W3C, а также из статей [12–14]. Важно заметить, что предыдущие анализы безопасности WebAuthn показывали его достаточно высокую криптостойкость в рамках стандарта FIDO2 [15, с. 453]. Принцип работы протокола CTAP2.1 был получен из официальной документации FIDO, а также из статей [16, 17]. В данной статье он не затрагивается так как не подвержен атаке, но он является частью стандарта. Отдельно рассматривается уязвимость, которую возможно осуществить на текущую версию протокола WebAuthn. Данная работа рассматривает уязвимость с точки зрения совокупности недостатков таких как WebAuthn, который является JavaScript API, и уязвимости, связанной с архитектурой стандарта, в которой присутствует возможность использования нелегитимных программ в качестве клиента. Данная атака реализовывалась автором самостоятельно, на момент написания статьи автору не попадались статьи, описывающие данную уязвимость.

Цель данного исследования заключается в подтверждение отсутствия доказательной устойчивости у текущего стандарта беспарольной аутентификации FIDO2 для последующей разработки безопасного протокола беспарольной аутентификации с использованием технологии Passkey [18, с. 203] на основе выявленных уязвимостей.

Описание стандарта FIDO2

Объектом исследования является стандарт беспарольной аутентификации FIDO2. В рамках работы FIDO2 лежат алгоритмы хеширования и принцип ассиметричного шифрования. Стойкие алгоритмы хеширования являются залогом стойкости криптосистем и текущий стандарт в этом случае не исключение. В источнике [19] первая глава посвящена алгоритмам хеширования.

В рамках работы стандарта FIDO2 используется ассиметричная криптография в варианте электронной подписи. Для этого открытый ключ отправляется в сервис и сохраняется на его сервере. Закрытый ключ надёжно хранится на пользовательском устройстве аутентификации (токене). Основным способом взаимодействия пользователя и аутентификатора является Authorization Gesture [20] или «Разрешающий жест» или «Жест авторизации». Под данным понятием подразумевается физическое взаимодействие пользователя с устройством аутентификации в рамках таких процедур как регистрация и аутентификация. Выполняя данный «жест», пользователь подтверждает, что разрешает выполнение процедуры. Данное действие может включать проверку пользователя, если используемое устройство аутентификации способно на это. Например, биометрическая

3 Тюрин К. А., Сёмин Р. В. Анализ стойкости парольных фраз на основе информационной энтропии // Известия ЮФУ. Технические науки. 2015 – No 5. – С. 18–27.

4 Chun Wang, Steve T.K. Jan, Hang Hu, Douglas Bossart, Gang Wang. The Next Domino to Fall: Empirical Analysis of User Passwords across Online Services // CODASPY '18: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy. 2018. – pp. 196-203. – DOI:10.1145/3176258.3176332.

проверка (автоматическое распознавание индивидов на основе их биологических и поведенческих характеристик) или простой ввод PIN-кода, или может включать простую проверку присутствия пользователя (простое нажатие на кнопку на аутентификаторе). Данный стандарт состоит из двух протоколов: WebAuthn и CTAP2.1.

Протокол веб-аутентификации

WebAuthn – это интерактивный протокол между тремя сторонами: токеном (ассоциируется с пользователем, который выполняет «жест авторизации»), клиентом (в рамках данного протокола это может быть приложение или веб-сервис) и сервером. Токен связан с открытым ключом подтверждения, который предварительно зарегистрирован на сервере. Протокол определяет два типа взаимодействий: регистрацию и аутентификацию. В рамках данной статьи рассмотрим лишь аутентификацию, которая и подвергается атаке.

Формальная схема процесса работы протокола WebAuthn во время выполнения аутентификации представлена на рисунке 1.

Во время выполнения процесса аутентификации используются следующие алгоритмы: Achallenge, Acommand, Aresponse, Acheck.

Инициация аутентификации происходит на стороне сервера алгоритмом Achallenge. В ходе его работы снова создается случайная строка битов rs длиной λ или более. Сервер не передается клиенту (браузеру) по каналу связи идентификатор пользователя uid, который был создан ранее, во время регистрации пользователя. В итоге клиенту отправляется сообщение cr, содержащее в себе идентификатор сервера idS и rs.

На стороне клиента происходит проверка идентификатора сервера и разрыв связи в случае несоответствия. В итоге работы алгоритма Acommand происходит формирование сообщения аутентификации Ma, в котором содержится идентификатор сервера и хешированная случайная строка, полученная от сервера.

На стороне токена алгоритм Aresponse извлекает учетные данные, связанные с идентификаторами сервера, из контекста регистрации токена. Затем увеличивает счетчик n на единицу. Далее происходит формирование подписи и ответного сообщения, а также обновление регистрационного состояния токена.

В конце на сервере запускается алгоритм Acheck, который по идентификатору учетных данных cid извлекает контекст регистрации на сервере. После этого происходит проверка если проверка успешна, то регистрационное состояние обновляется на сервере.

Атака на протокол WebAuthn

В рамках основной задачи проводилось исследования протокола и его формализованной формы на наличие уязвимостей. В ходе анализа, а также изучения информации из свободных источников была обнаружена потенциальная уязвимость, которая позволяет получить доступ к учетной записи с включённой беспарольной аутентификацией по стандарту FIDO2. Данная уязвимость подтверждает отсутствие доказательной устойчивости у текущего стандарта. В качестве аутентификатора (токена) выступал персональный компьютер с включённым windows hello. Запрос осуществляется в рамках аутентификации на веб-сервисе.

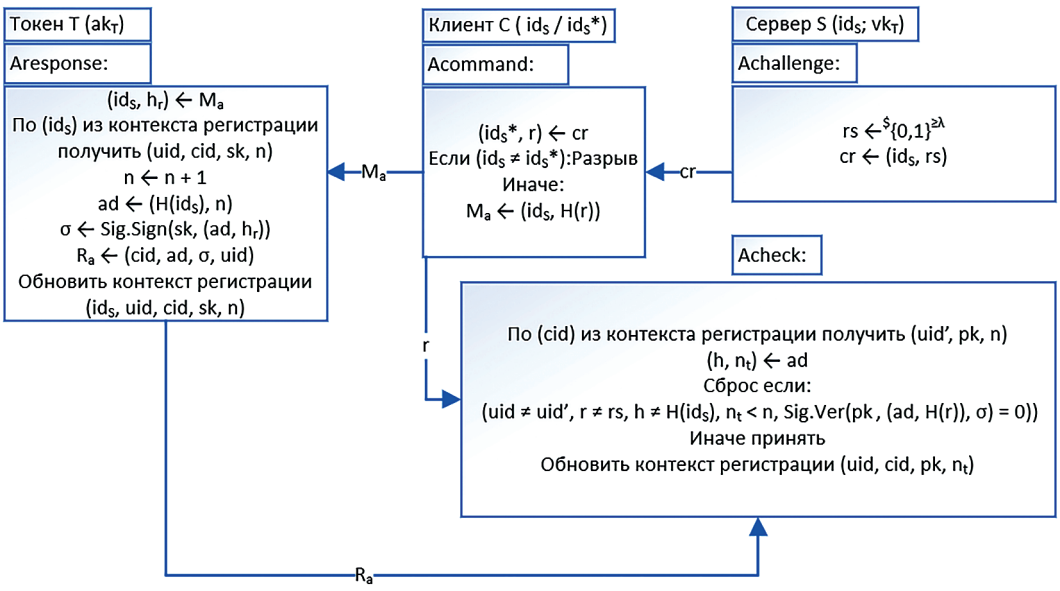

Рис. 1. Протокол WebAuthn (аутентификация)

В рамках выполнения данной атаки нужно обговорить несколько важных моментов и только после этого перейти к рассмотрению сценария атаки, его реализации и результатам её выполнения.

Данная атака проводится на уже зарегистрированного пользователя. Сам пользователь мог ранее быть зарегистрирован на основании классического логина и пароля, но позже к его аккаунту была подключена и беспарольная аутентификация на основе стандарта FIDO2. Проверяющая сторона (RP) осуществляет проверку пользователя на основе криптографических подходов – цифровой подписи. В рамках стандарта, если пользователь зарегистрирован сразу беспарольными способом или включил его позднее, можно сказать, что у него уже есть пара криптографических ключей. Открытый ключ и учетные данные уже есть на сервере RP, а закрытый ключ никогда не покидает токен пользователя. При аутентификации пользователь запрашивает сервер RP, который извлекает идентификаторы учетных данных пользователя, запросившего аутентификацию. После этого формируются параметры, которые должны быть переданы клиенту (браузеру) пользователя, через протокол https в ответ на вызов API (WebAuthn). Далее в рамках работы протокола WebAuthn происходит проверка идентификатора сервера, от имени которого пришли параметры. Если все верно, то происходит вызов функции navigator.credentials.get(). Данная функция передает информацию по протоколу CTAP2.1 на аутентификатор. Обычно для подписи запроса используется алгоритм ECDSA[5,6].

Оценка уязвимости

Нужно понимать, что веб-аутентификация это JavaScript API, это значит, что WebAuthn будет следовать всем правилам работы данного языка программирования. А именно переопределению функций[7]. Переопределение функции в JavaScript — это способность дочернего класса или подкласса предоставить реализацию конкретного метода, который уже определён в родительском классе или его суперклассе. В данном случае это подразумевает, что можно запустить пользовательскую реализацию функции navigator.credentials.get() в виде «межсайтового скриптинга» (XSS) [20]. Данное действие должно переопределить собственную реализацию этой же функции в оригинальном WebAuthn. В итоге вызов WebAuthn не будут функционировать так, как это задумывалось.

5 Michael Braun, Anton Kargl. A Note on Signature Standards // IACR Cryptology ePrint Archive 2007/357, 2007.

6 Johnson, D., Menezes, A. & Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA). IJIS 1, 36–63 (2001). https://doi.org/10.1007/s102070100002.

7 Bloch Joshua, Guy L.Steele, Jr. Effective Java: programming language guide. – edition 2. print. May 2001. – 180 pp.

Пример такого переопределения показан на рисунке 2.

```
> class cred
  {
  static get(options)
  {
  console.log("Пример переопределения функции");
  }
  }
  class navigator
  {
  }
  navigator.credentials=cred;
< class cred
  {
  static get(options)
  {
  console.log("Пример переопределения функции");
  }
  }
  Пример переопределения функции                    VM3937:5
```

Рис. 2. Пример переопределения функции

В рамках атаки злоумышленник может выполнить «самостоятельный межсайтовый скриптинг» (Self-XSS) в своей собственной системе, изменив функциональность WebAuthn для себя. В результате, мера защиты, которую, например, Google использует на странице входа в свой аккаунт, становится бесполезной, так как предупреждение видит только злоумышленник, совершающий это намеренно. Предупреждение показано на рисунке 3.

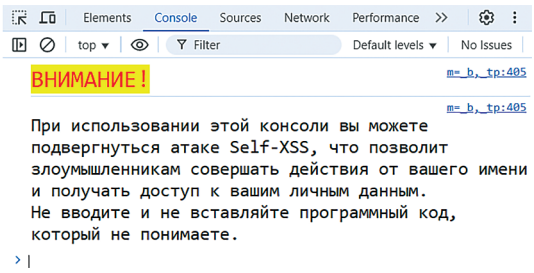

Рис. 3. Предупреждение в консоли на официальной странице google

Сценарий атаки

Рассмотрим сценарий самой атаки.

1. Злоумышленник должен развернуть сервер, который будет находиться в том же участке сети, что и легитимный пользователь. Данный сервер будет осуществлять все пересылку между злоумышленником и вредоносным ПО на компьютере жертвы. Важно заметить, что для попадания в локальную сеть жертвы могут использоваться недостатки или уязвимости из других веб-приложений. «По данным компании Positive Technologies 17 % от общего числа атак связаны с уязвимостями и недостатками защиты веб-приложений, которые могут быть использованы для проникновения в локальный сетевой периметр организации или распространения вредоносного программного обеспечения» [21].

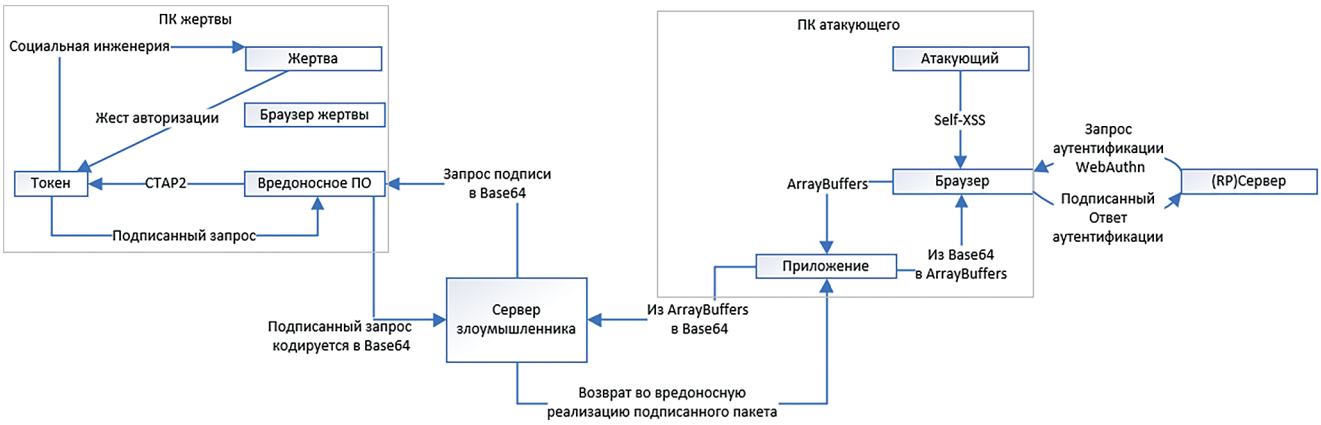

Рис. 4. Сценарий атаки

2. Злоумышленник запускает приложение, который копирует в буфер обмена код нужной ему Self-XSS. Этот код далее необходимо вставить в консоль браузера перед вызовом WebAuthn. После того, как код будет вставлен произойдет переопределение navigator.credentials.get(), параметры из которой будут считаны приложение, запущенным на машине злоумышленника. Эти данные будут закодированы в формате Base64 и после этого будут пересланы на облачный сервер.

3. Вредоносным ПО в системе жертвы запрашивает у сервера данные в формате Base64. Затем декодирует данные в формат, который соответствует WebAuthn. После чего запрашивает токен для подписания данного запроса. Тут в дело вступает социальная инженерия, о которой уже говорил ранее. После того как запрос подписан он заново кодируется и оправляется обратно на сервер. Сам сервер после этого оправит ответ на систему атакующего, точнее на его приложение. Нужно добавить небольшую ремарку. Способ попадания вредоносной программы может быть самым разнообразным. «По статистике, веб-браузер является одним из основных инструментов доставки вредоносных программ на компьютеры пользователей» [22]. Потенциальным способом защиты в этом случае может быть фаззинг-тестирование. В статье [22] рассмотрены разные подходы к фаззингу. Для передачи вредоносного ПО на систему жертвы может использоваться DNS-туннелирование. В статье [23] отражены основные особенности работы DNS-туннелирования как способа доставки вредоносного ПО на узел, защищенный межсетевым экраном.

4. Далее приложение декодирует Base64 и возвращает его вредоносной реализации navigator.credentials.get() которую запускали при помощи Sefl-XSS из буфера обмена в самом начале атаки. После этого вредоносная реализация декодирует его обратно в ArrayBuffers и возвращает на сервер RP.

Сценарий атаки в виде схемы показан на рисунке 4.

Выполнение атаки

В рамках ранее описанного сценария проведем выполнение атаки и зафиксируем ее результаты. Первое, что нужно сделать – развернуть сервер, который будет пересылать сообщения между участниками атаки. Данный сервер является основным в данной атаке, но функционал его достаточно прост – пересылка сообщений между ПК атакующего и ПК жертвы. Он, как и все остальные компоненты эксплойта,

Рис. 5. Схема работы сервера между ПК атакующего и жертвы

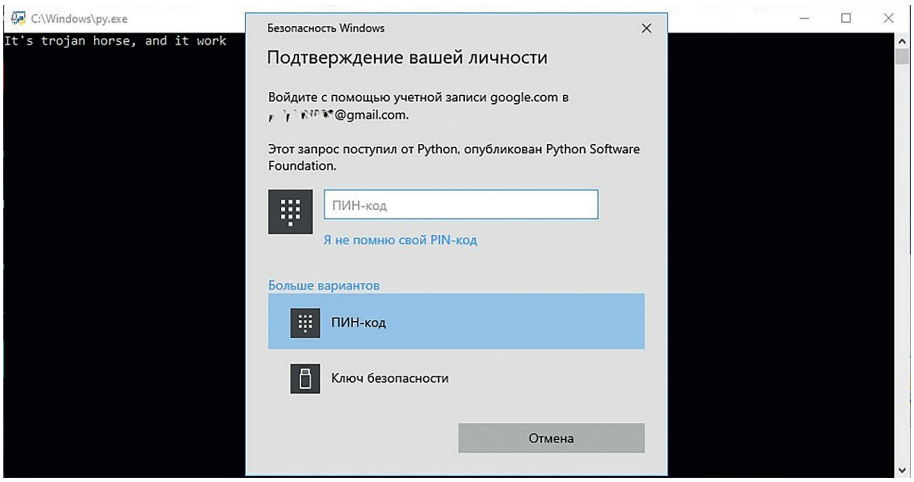

Рис. 6. Запрос «жеста авторизации»

был реализован автором. Сам сервер был реализован на основе Apache2 с использованием mod_wsgi, для использования скрипта, написанного на python с использованием библиотеки flask и расширением CORS. Пример блок-схемы работы сервера показан на рисунке 5.

Далее злоумышленник запускает на своей стороне программу, которая должна будет считать данные от вредоносной реализации. По понятным соображениям публиковать код эксплойта автор не должен, но принцип работы объяснен будет. Данную реализацию посредством Self-XSS запускает сам атакующий. Важно уточнить, что сделать это нужно перед запуском работы протокола WebAuthn.

Данные, полученные от вредоносной реализации функции WebAuthn, будут закодированы в нужный формат и пересланы на сервер, который уже был ранее запущен. После этого сервер отправит закодированные данные на вредоносное ПО в системе жертвы. Данное ПО декодирует данные в формат необходимый для подписи и, имитируя из себя клиента, сделает запрос к аутентификатору. В рамках работы протокола CTAP2.1 будет выведен запрос на совершение «жеста авторизации», тут жертва, поддавшаяся атаке социальной инженерии, введет свой pin-код. Пример запроса «жеста авторизации» токена после запроса от вредоносного программного обеспечения типа троянский конь, показана на рисунке 6.

После того как будет выполнен «жест авторизации» и запрос будет подписан, вредоносное ПО закодирует запрос в нужный формат и отправит его обратно на сервер. Код по понятным причинам опубликован быть не может, поэтому будет приведена блок-схема алгоритма работы вредоносного ПО (рис. 7).

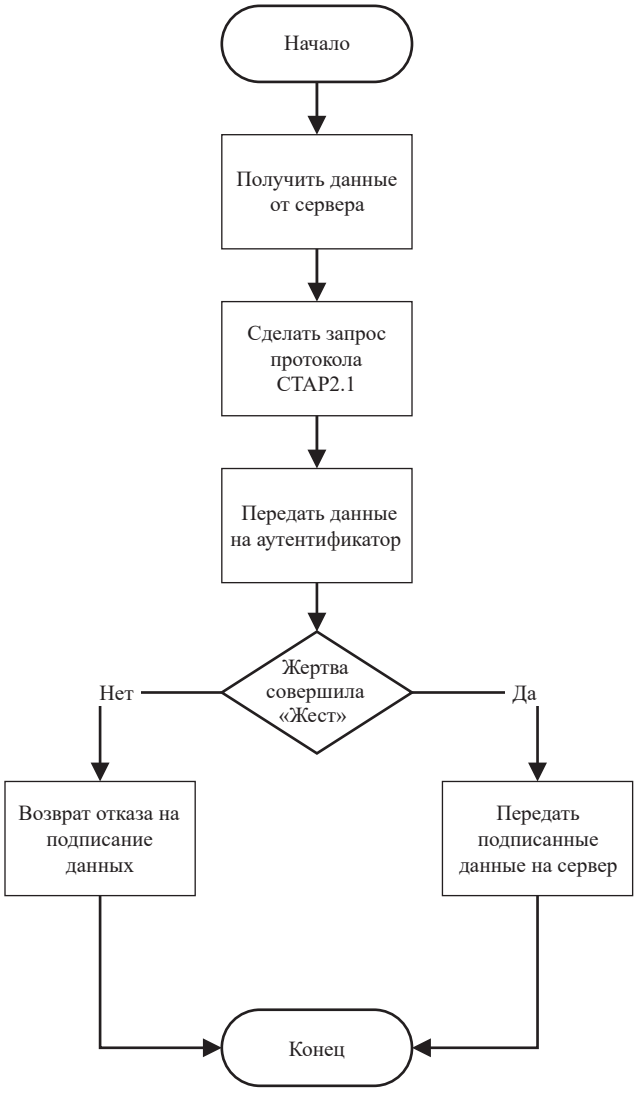

Рис. 7. Схема работы вредоносного ПО

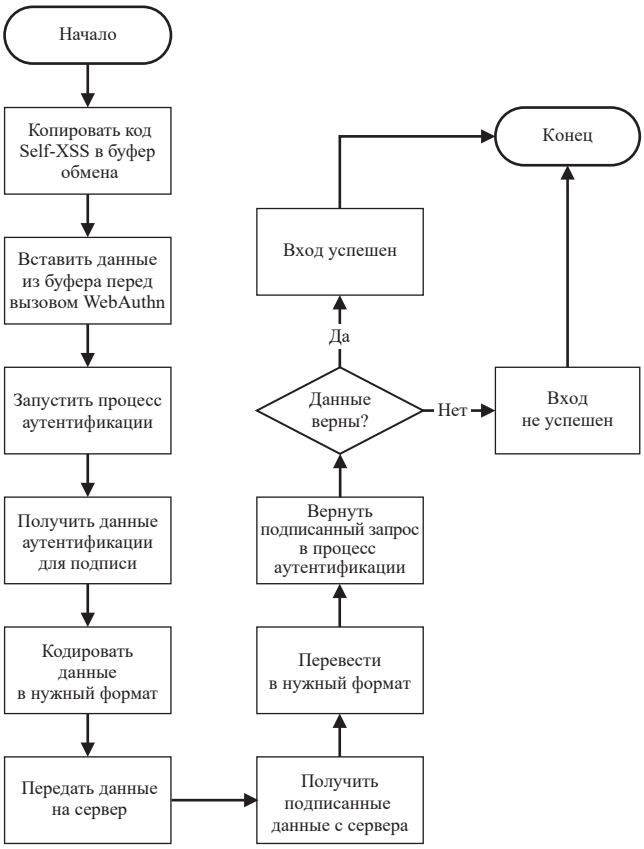

Рис. 8. Схема работы программы на ПК атакующего

После этого приложение на компьютере атакующего получит данные с сервера, переведет их в нужный формат и вернет во вредоносную реализацию, которая ранее была запущена. В конце вредоносная реализация вернет данные в формат WebAuthn и отправит их на сервер RP. После этого сервер проверит подпись и убедившись в том, что ключ и подпись соответствуют тем, что были с момент регистрации легитимного пользователя, предоставит доступ. Пример блок-схемы алгоритма работы программы на ПК атакующего показан на рисунке 8.

В итоге, в рамках экспериментов, был получен доступ к учетной записи Google, которая использовала беспарольную аутентификацию. Подтверждение успешной аутентификации показан на рисунке 9. Код 200 означает успех.

Потенциальные способы защиты

На данный момент рассматривается несколько потенциальных вариантов, если не закрытия данной уязвимости, то хотя бы усложнения ее реализации. Основой атаки выступает возможность переопределения функции в языке JavaScript. Это означает, что большинство браузеров подвержены данной уязвимости. Предлагаемое автором решение не является истинной в последней инстанции, а лишь рекомендация по улучшению реализации WebAuthn. При длительном рассмотрение будет найдены способы обхода данного изменения, но данное изменение позволяет усложнить подготовку атаки. Пример уязвимой реализации WebAuthn показан на рисунке 10.

На девятнадцатой строке вызывается navigator. credentials.get(), которая может быть переопределена злоумышленником за счет Self-XSS. Пример такого переопределения ранее был показан на рисунке 2. Способ защиты следующий. Известно, что скрипты, которые находятся в теге head выполняется в первую очередь. И не асинхронные функции в теге head всегда будут выполняться до вызова любой другой функции onload или до выполнения Self-XSS в консоли. Эта особенность была бы полезна для устранения уязвимости, описанной в данной статье. В рекомендуемой реализации WebAuthn можно предложить разработчику создать некую

```
[02/Mar/2025 20:09:02] "OPTIONS /getoptions?site=https://accounts.google.com HTTP/1.1" 200 -
[02/Mar/2025 20:09:18] "POST /getoptions?site=https://accounts.google.com HTTP/1.1" 200 -
```

Рис. 9. Ответ сервера RP

```
11  <script>
12    function start()
13    fetch('/api/authenticate/begin' {,
14      method: "POST",
15    }).then(function(response) {
16      if(response.ok) return response. arrayBuffer();
17      throw new Error('No credential available to authenticate! ');
18    }).then(CBOR. decode). then(function(options) {
19      return navigator.credentials.get(options);
20    }).then ( function(assertion) {
21      return fetch('/api/authenticate/complete', {
22        method: "POST",
23        headers: ["Content-Type": "application/cbor"
24        body: CBOR. encode(f
25      "credentialId": new Uint8Array(assertion.rawId),
26      "authenticatorData": new Uint8Array(assertion. response. authenticatorData),
27      "clientDataJSON": new Uint8Array(assertion. response. clientDataJSON) ,
28      "signature": new Uint8Array(assertion. response. signature)
29    })
```

Рис. 10. Пример уязвимой реализации WebAuthn

```
function start()
fetch('/api/authenticate/begin' {,
  method: "POST",
}).then(function(response) {
  if(response.ok) return response. arrayBuffer();
  throw new Error('No credential available to authenticate! ');
}).then(CBOR. decode). then(function(options) {
  return unknowntotheattackerfunction(options);
}).then ( function(assertion) {
  return fetch('/api/authenticate/complete', {
    method: "POST",
    headers: ["Content-Type": "application/cbor"
    body: CBOR. encode(f
  "credentialId": new Uint8Array(assertion.rawId),
  "authenticatorData": new Uint8Array(assertion. response. authenticatorData),
  "clientDataJSON": new Uint8Array(assertion. response. clientDataJSON) ,
  "signature": new Uint8Array(assertion. response. signature)
```

Рис. 11. Модифицированная версия реализации WebAuthn

резервную копию функции navigator.credentials.get() с таким именем, которое может быть неизвестно для злоумышленника. Данное имя нужно сделать или динамическим, или случайным. Или данное имя может быть скрыто, чтобы злоумышленник не знал его. Последний вариант возможно сделать, с помощью такой строки кода: const unknowntotheattackerfunction =navigator.credentials.get.bind(navigator.credentials), расположенной в теге head, тем самым зафиксировав её выполнение до консоли.

В основной части веб-страницы вместо «navigator. credentials.get()» нужно использовать вызов функции «unknowntotheattackerfunction()». Пример модифицированной версии реализации WebAuthn показан на рисунке 11.

Данная модификация позволит усложнить реализацию атаки так как указатель на исходную функцию уже находится в браузере и не может быть переопределен, если имя скрыто. Данная защита не панацея, а лишь усложнение атаки для злоумышленника.

Отдельно можно рассмотреть скорее не способ защиты, а способ предупреждения пользователя. Так как текущая версия стандарта FIDO2 сильно полагается на «жест авторизации», то необходимо расширить данные, которые собираются на этапе регистрации, а затем отправляются на подпись пользователю при входе. Нужно фиксировать «fingerprint» браузера, с которого происходила регистрация и в момент входа сверять его с тем, что хранится на сервере. Если они разные, то нужно записать в «параметры» функции navigator.credentials.get() флаг с тем, что данные не совпали и это потенциально не легитимный пользователь. Дальше данный флаг будет обработан токеном и будет выдано предупреждение: «Обнаружен вход с неизвестного устройства. Если это не вы, то прервите текущие операции входа!» Данная атака не затрагивает протокол CTAP2.1, а лишь социальной инженерией заставляет пользователя поверить в то, что вход осуществляет он. И пользователь не видит никаких предупреждений и может не насторожиться.

Заключение

В настоящей работе проведено исследование текущего стандарта беспарольной аутентификации FIDO2 на наличие уязвимостей. В рамках обнаруженной уязвимости было проведена разработка, развертывание всех необходимых компонентов. После чего экспериментально была подтверждена возможность эксплуатации данной уязвимости на актуальной версии стандарта FIDO2. Данная уязвимость позволяет заявить о возможности несанкционированное доступа к учетным данным пользователя во время процесса аутентификации. Уязвимость оказывает влияет на безопасность систем аутентификации.

В рамках данного исследования было подтверждено отсутствие доказательной устойчивости у текущего стандарта. Обнаруженная уязвимость позволяет получить доступ к аккаунту, защищенному по текущему стандарту беспарольной аутентификацией. При реализации данной уязвимости сложностью для атакующего в ней является заражение компьютера жертвы вредоносным программным обеспечением типа троянский конь и последующая атака социальной инженерии.

В рамках стандарта беспарольной аутентификации FIDO2 считается, что «жест авторизации», который делает пользователь, и информация, которая отобразится на токене, позволит человеку отличить настоящий запрос, сделанный им, от мошеннической схемы. Данное предположение можно было бы считать верными, если бы не статистика успешных мошеннических атак по всему миру за последние несколько лет. Есть предложения касательно способа потенциальной защиты от данной уязвимости, а также способ сбора дополнительных данных на этапе регистрации для уведомления пользователя в процессе атаки. Все данные способы, с точки зрения автора, скорее временные заплатки. Для полноценной защиты нужно рассматривать другую архитектуру для будущего протокола, которую автор надеется предложить в будущем.

Литература

1. Современные направления применения комбинаторики в области защиты персональных данных / М. О. Тенячкина, М. В. Богданова, К. И. Быкова, К. А. Сакалова // Международный научно-исследовательский журнал. – 2024. – № 9(147). – DOI: 10.60797/IRJ.2024.147.1.
2. Салита, Д. С. Методы оценки надежности парольных систем / Д. С. Салита, А. А. Удовик // Проблемы правовой и технической защиты информации. – 2020. – № 8. – С. 47–51.
3. Информационная безопасность: парольная защита / И. А. Рябов, М. Г. Койцан, А. А. Кузнецов, В. В. Ермолаева // Тенденции развития науки и образования. – 2023. – № 93-8. – С. 76–79. – DOI 10.18411/trnio-01-2023-401.
4. Назаров, Д. М. Методика создания надежного пароля для обеспечения экономической безопасности в условиях цифровизации / Д. М. Назаров // Известия Санкт-Петербургского государственного экономического университета. – 2022. – № 1(133). – С. 155–160.
5. Кочанова А. Г. Надёжные пароли: как их создать и чем они полезны // Вестник науки. 2023. № 6(63).
6. Абидарова А. А. Анализ надежности паролей для обеспечения информационной безопасности // Известия Тульского государственного университета. Технические науки. 2021. – № 8. – С. 66–68.
7. Селиверстов В. В, Корчагин С. А. Анализ актуальности и состояния современных фишинг-атак на объекты критической информационной инфраструктуры // Инженерный вестник Дона. 2024. – № 6. – С. 216–229.
8. Степкин, Б. А. Как требования к паролю влияют на его безопасность / Б. А. Степкин, С. В. Малахов // Скиф. Вопросы студенческой науки. – 2021. – № 4(56). – С. 83–86.
9. Lyastani, S. G., Schilling, M., Neumayr, M., Backes, M., & Bugiel, S. (2020). Is FIDO2 the kingslayer of user authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. 2022 IEEE Symposium on Security and Privacy (SP), 268–285. https://doi.org/10.1109/sp40000.2020.00047
10. Докучаев В А., Мытенков С. С., Рахмани Д. Д., Сафонов И. А. Анализ уязвимостей и рисков традиционных парольных систем в контексте корпоративных распределенных систем и критически важных инфраструктур // Экономика и качество систем связи. 2025. № 36. С. 135–147.
11. Mitra, A., & Ghosh, A. (2024). FIDO2: A comprehensive study on passwordless authentication. International Journal of Engineering Research and Applications, 14(7), 58–63. https://doi.org/10.9790/9622-14075863.
12. Dourado, M. R., Gestal, M., & Vázquez-Naya, J. M. (2020). Implementing a web application for W3C WebAuthn protocol testing. MDPI, 5. https://doi.org/10.3390/proceedings2020054005.
13. Bindel, N., Gama, N., Guasch, S., Ronen, E. (2023). To Attest or Not to Attest, this is the Question – Provable Attestation in FIDO2. In: Guo, J., Steinfeld, R. (eds) Advances in Cryptology – ASIACRYPT 2023. ASIACRYPT 2023. Lecture Notes in Computer Science, vol. 14443. Springer, Singapore. https://doi.org/10.1007/978-981-99-8736-8_10.
14. Hanzlik, L., Loss, J., & Wagner, B. (2023). Token meets Wallet: Formalizing Privacy and Revocation for FIDO2. 2022 IEEE Symposium on Security and Privacy (SP), 1491–1508. https://doi.org/10.1109/sp46215.2023.10179373.
15. Gudipati, R. R. (2025). Demystifying fido: a technical deep dive into modern authentication // International journal of information technology and management information systems, 16(2), 452–466. https://doi.org/10.34218/ijitmis_16_02_029.
16. Bindel, N., Cremers, C., & Zhao, M. (2023b). FIDO2, CTAP 2.1, and WebAuthN 2: Provable Security and Post-Quantum Instantiation. 2022 IEEE Symposium on Security and Privacy (SP). https://doi.org/10.1109/sp46215.2023.10179454.
17. Guan, J., Li, H., Ye, H., Zhao, Z. (2022). A Formal Analysis of the FIDO2 Protocols. In: Atluri, V., Di Pietro, R., Jensen, C. D., Meng, W. (eds) Computer Security – ESORICS 2022. ESORICS 2022. Lecture Notes in Computer Science, vol 13556. Springer, Cham. https://doi.org/10.1007/978-3-031-17143-7_1.
18. Dr. A.Shaji George. (2024). The Dawn of Passkeys: Evaluating a Passwordless Future. Partners Universal Innovative Research Publication (PUIRP), 02(01), 202–220. https://doi.org/10.5281/zenodo.10697886.
19. Криптографические основы блокчейн-технологий / Е. А. Ищукова, С. П. Панасенко, К. С. Романенко, В. Д. Салманов. – Москва: ООО «ДМК Пресс. Электронные книги», 2022. – 301 с.
20. Barbosa, M., Boldyreva, A., Chen, S., Warinschi, B. (2021). Provable Security Analysis of FIDO2. In: Malkin, T., Peikert, C. (eds) Advances in Cryptology – CRYPTO 2021. CRYPTO 2021. Lecture Notes in Computer Science(), vol 12827. Springer, Cham. https://doi.org/10.1007/978-3-030-84252-9_5.
21. Боровков В. Е. Методы защиты веб-приложений от злоумышленников / В. Е. Боровков, П. Г. Ключарев // Вопросы кибербезопасности. – 2023. – № 5(57). – С. 89–99. – DOI 10.21681/2311-3456-2023-5-89-99.
22. Козачок, А. В. Подходы к оценке поверхности атаки и фаззингу веб-браузеров / А. В. Козачок, Д. А. Николаев, Н. С. Ерохина // Вопросы кибербезопасности. – 2022. – № 3(49). – С. 32–43. – DOI 10.21681/2311-3456-2022-3-32-43.
23. Москвичев А. Д. Использование DNS-туннелирования для передачи вредоносного программного обеспечения / А. Д. Москвичев, К. С. Москвичева // Вопросы кибербезопасности. – 2022. – № 4(50). – С. 91–99. – DOI 10.21681/2311-3456-2022-4-91-99.

USES OF WEBAUTHN PROTOCOL VULNERABILITIES TO OBTAIN UNSANCTIONED ACCESS

Panchenko A. R.[8]

Keywords: *password-free authentication; asymmetric encryption; Self-XSS; Python; FIDO2; CTAP2.1; Authorization Gesture.*

Purpose of the study: *investigation of the current FIDO2 password-free authentication standard for vulnerabilities and confirmation of its lack of evidence-based stability.*

Methods of research: *analysis of the current password-free authentication standard for vulnerabilities. Implementation and exploitation of the discovered vulnerability. This vulnerability is based on the use of a modified man-in-the-middle attack using malicious software and social engineering.*

8 Alexander R. Panchenko, postgraduate student, Southern Federal University «SFedU», Institute of Computer Technologies and Information Security, Taganrog, Russia. ORCID: https://orcid.org/0009-0001-4720-5164. E-mail: alpanchenko@sfedu.ru

Research results: *the analysis of the current standard of password-free authentication is carried out. As part of this analysis, a formal scheme of the WebAuthn protocol was formed. A vulnerability has been discovered that allows access to the account of a legitimate user protected by the current FIDO2 standard. To implement this vulnerability, a modified man-in-the-middle attack was used, in which a server was implemented that transmits messages during the attack. Malicious Trojan horse type software was implemented. This malicious software was previously placed in the victim's system, where it posed as the victim's browser, which was contacted by a legitimate server. On the attacker's side, a program was implemented that reads the authentication request in the attacker's system. This program is used to override the function during the authentication process and then forward the request to the server. As part of the implementation of this vulnerability and its subsequent implementation, the lack of evidence-based stability of the current FIDO2 password-free authentication standard was confirmed. A method of potential protection against this vulnerability has been proposed. A method was also proposed to modify data collection at the registration stage, for subsequent notification of a potential victim attack during the signing procedure.*

Scientific novelty: *the discovered vulnerability confirms the lack of evidence-based stability of the current FIDO2 password-free authentication standard. Currently, all browsers use WebAuthn, which means that all modern browsers are affected by this vulnerability. A potential way to protect against this vulnerability and a way to prevent the victim from being attacked is proposed.*

References

1. Sovremennye napravlenija primenenija kombinatoriki v oblasti zashhity personal'nyh dannyh / M. O. Tenjachkina, M. V. Bogdanova, K. I. Bykova, K. A. Sakalova // Mezhdunarodnyj nauchno-issledovatel'skij zhurnal. – 2024. – № 9(147). – DOI: 10.60797/IRJ.2024.147.1.
2. Salita, D. S. Metody ocenki nadezhnosti parol'nyh sistem / D. S. Salita, A.A. Udovik // Problemy pravovoj i tehnicheskoj zashhity informacii. – 2020. – № 8. – S. 47–51.
3. Informacionnaja bezopasnost': parol'naja zashhita / I. A. Rjabov, M. G. Kojcan, A.A. Kuznecov, V. V. Ermolaeva // Tendencii razvitija nauki i obrazovanija. – 2023. – № 93-8. – S. 76–79. – DOI 10.18411/trnio-01-2023-401.
4. Nazarov, D. M. Metodika sozdanija nadezhnogo parolja dlja obespechenija jekonomicheskoj bezopasnosti v uslovijah cifrovizacii / D. M. Nazarov // Izvestija Sankt-Peterburgskogo gosudarstvennogo jekonomicheskogo universiteta. – 2022. – № 1(133). – S. 155–160.
5. Kochanova A. G. Nadjozhnye paroli: kak ih sozdat' i chem oni polezny // Vestnik nauki. 2023. № 6(63).
6. Abidarova A. A. Analiz nadezhnosti parolej dlja obespechenija informacionnoj bezopasnosti // Izvestija Tul'skogo gosudarstvennogo universiteta. Tehnicheskie nauki. 2021. – № 8. – S. 66–68.
7. Seliverstov V. V, Korchagin S. A. Analiz aktual'nosti i sostojanija sovremennyh fishing-atak na ob#ekty kriticheskoj informacionnoj infrastruktury // Inzhenernyj vestnik Dona. 2024. – № 6. – S. 216–229.
8. Stepkin, B. A. Kak trebovanija k parolju vlijajut na ego bezopasnost' / B. A. Stepkin, S. V. Malahov // Skif. Voprosy studencheskoj nauki. – 2021. – № 4(56). – S. 83–86.
9. Lyastani, S. G., Schilling, M., Neumayr, M., Backes, M., & Bugiel, S. (2020). Is FIDO2 the kingslayer of user authentication? A Comparative Usability Study of FIDO2 Passwordless Authentication. 2022 IEEE Symposium on Security and Privacy (SP), 268–285. https://doi.org/10.1109/sp40000.2020.00047
10. Dokuchaev V A., Mytenkov S. S., Rahmani D. D., Safonov I. A. Analiz ujazvimostej i riskov tradicionnyh parol'nyh sistem v kontekste korporativnyh raspredelennyh sistem i kriticheski vazhnyh infrastruktur // Jekonomika i kachestvo sistem svjazi. 2025. № 36. S. 135-147.
11. Mitra, A., & Ghosh, A. (2024). FIDO2: A comprehensive study on passwordless authentication. International Journal of Engineering Research and Applications, 14(7), 58–63. https://doi.org/10.9790/9622-14075863.
12. Dourado, M. R., Gestal, M., & Vázquez-Naya, J. M. (2020). Implementing a web application for W3C WebAuthn protocol testing. MDPI, 5. https://doi.org/10.3390/proceedings2020054005.
13. Bindel, N., Gama, N., Guasch, S., Ronen, E. (2023). To Attest or Not to Attest, this is the Question – Provable Attestation in FIDO2. In: Guo, J., Steinfeld, R. (eds) Advances in Cryptology – ASIACRYPT 2023. ASIACRYPT 2023. Lecture Notes in Computer Science, vol 14443. Springer, Singapore. https://doi.org/10.1007/978-981-99-8736-8_10.
14. Hanzlik, L., Loss, J., & Wagner, B. (2023). Token meets Wallet: Formalizing Privacy and Revocation for FIDO2. 2022 IEEE Symposium on Security and Privacy (SP), 1491–1508. https://doi.org/10.1109/sp46215.2023.10179373.
15. Gudipati, R. R. (2025). Demystifying fido: a technical deep dive into modern authentication // International journal of information technology and management information systems, 16(2), 452–466. https://doi.org/10.34218/ijitmis_16_02_029.
16. Bindel, N., Cremers, C., & Zhao, M. (2023b). FIDO2, CTAP 2.1, and WebAuthN 2: Provable Security and Post-Quantum Instantiation. 2022 IEEE Symposium on Security and Privacy (SP). https://doi.org/10.1109/sp46215.2023.10179454.
17. Guan, J., Li, H., Ye, H., Zhao, Z. (2022). A Formal Analysis of the FIDO2 Protocols. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds) Computer Security – ESORICS 2022. ESORICS 2022. Lecture Notes in Computer Science, vol 13556. Springer, Cham. https://doi.org/10.1007/978-3-031-17143-7_1.
18. Dr. A.Shaji George. (2024). The Dawn of Passkeys: Evaluating a Passwordless Future. Partners Universal Innovative Research Publication (PUIRP), 02(01), 202–220. https://doi.org/10.5281/zenodo.10697886.
19. Kriptograficheskie osnovy blokchejn-tehnologij / E. A. Ishhukova, S. P. Panasenko, K. S. Romanenko, V. D. Salmanov. – Moskva: OOO «DMK Press. Jelektronnye knigi», 2022. – 301 s.
20. Barbosa, M., Boldyreva, A., Chen, S., Warinschi, B. (2021). Provable Security Analysis of FIDO2. In: Malkin, T., Peikert, C. (eds) Advances in Cryptology – CRYPTO 2021. CRYPTO 2021. Lecture Notes in Computer Science(), vol 12827. Springer, Cham. https://doi.org/10.1007/978-3-030-84252-9_5.
21. Borovkov V.E. Metody zashhity veb-prilozhenij ot zloumyshlennikov / V. E. Borovkov, P. G. Kljucharev // Voprosy kiberbezopasnosti. – 2023. – № 5(57). – S. 89–99. – DOI 10.21681/2311-3456-2023-5-89-99.
22. Kozachok, A. V. Podhody k ocenke poverhnosti ataki i fazzingu veb-brauzerov / A. V. Kozachok, D. A. Nikolaev, N. S. Erohina // Voprosy kiberbezopasnosti. – 2022. – № 3(49). – S. 32-43. – DOI 10.21681/2311-3456-2022-3-32-43.
23. Moskvichev A. D. Ispol'zovanie DNS-tunnelirovanija dlja peredachi vredonosnogo programmnogo obespechenija / A. D. Moskvichev, K. S. Moskvicheva // Voprosy kiberbezopasnosti. – 2022. – № 4(50). – S. 91-99. – DOI 10.21681/2311-3456-2022-4-91-99.