Уязвимости экосистемы MCP

Д.Е. Намиот, Е.А. Ильюшин

Аннотация— **Протокол контекста модели (MCP) — это открытый стандарт, позволяющий разработчикам создавать безопасные двусторонние соединения между источниками данных и инструментами на базе ИИ. Архитектура, как заявляет компания Anthropic проста: разработчики могут либо предоставлять доступ к своим данным через серверы MCP, либо создавать приложения на основе ИИ (MCP-клиенты), подключающиеся к этим серверам. Протокол претендует на роль нового стандарта для подключения ИИ-агентов (AI-агенты или агенты с искусственным интеллектом) к системам хранения данных, включая репозитории контента, бизнес-инструменты и среды разработки. Агенты, по самому общему определению, представляют собой некоторые автономно работающие системы, которые используют методы искусственного интеллекта для достижения поставленных целей. Цель MCP - помочь передовым моделям выдавать более качественные и релевантные ответы. MCP должен стать основой для Интернета ИИ-агентов. Протокол быстро завоевал популярность благодаря простоте использования и преимуществам, которые он предоставляет при использовании ИИ. Вместе с тем, нужно сказать, что MCP был разработан, в первую очередь, исходя из идей функциональности, а не для безопасности. Его применение создает новые фундаментальные уязвимости и расширяет поверхности атак. Здесь соединяются риски генеративных моделей ИИ-агентов и уязвимости программного обеспечения экосистемы MCP. Уязвимостям экосистемы MCP к кибератакам и посвящена настоящая статья. Работа является продолжением серии публикаций, посвященных безопасности ИИ-агентов.**

***Ключевые слова*—кибербезопасность, LLM, MCP, агенты.**

I. Введение

Протокол контекста модели (MCP) — это протокол прикладного уровня от компании Anthropic, определяющий способ подключения больших языковых моделей (LLM) к внешним инструментам [1]. Компания Anthropic описывает его так: “Представьте себе порт USB-C для приложений ИИ. Подобно тому, как USB-C обеспечивает стандартизированный способ подключения устройств к различным периферийным устройствам и аксессуарам, MCP обеспечивает стандартизированный способ подключения моделей ИИ к различным источникам данных и инструментам.”

Протокол быстро завоевал популярность благодаря простоте использования и преимуществам, которые он предоставляет при использовании ИИ. В этой статье рассматриваются некоторые потенциальные риски безопасности, с которыми связана эксплуатация экосистемы MCP.

Статья получена 31 июля 2025. Д.Е. Намиот – МГУ имени М.В. Ломоносова (email: dnamiot@gmail.com)
Е.А. Ильюшин – МГУ имени М.В. Ломоносова (email: john.ilyushin@gmail.com)

Почему мы говорим об экосистеме? Это связано с тем, что у нас есть разные компоненты. MCP не связывает LLM с инструментами напрямую. Клиентский компонент MCP обращается к LLM, а серверный компонент MCP - к инструментам. Один клиент MCP имеет доступ к одному или нескольким серверам MCP. Пользователи могут подключать любое количество серверов MCP к клиенту MCP.

Вот как выглядит типичный обмен данными между клиентом и сервером MCP:

- Пользователь запрашивает задачу у клиента MCP.
- Клиент MCP располагает информацией об инструментах, которые использует или к которым имеет доступ каждый сервер MCP.
- Клиент MCP передаёт запрос пользователя и информацию с серверов MCP в LLM, который отвечает, предоставляя необходимый инструмент и параметры для использования.
- Клиент MCP отправляет информацию об инструментах и параметрах на сервер MCP.
- Сервер MCP завершает задачу и возвращает ответ клиенту MCP, который передаёт ответ LLM, а LLM генерирует ответ для пользователя.
- Клиент MCP отображает ответ пользователю.

Это проиллюстрировано на рис. 1.

Серверы MCP могут работать локально или удалённо, и риски безопасности различаются в зависимости от способа их работы.

Мы определяем «локальные серверы MCP» как серверы MCP, работающие на хосте, которым мы управляем. Для выполнения требуемых задач локальные серверы MCP обычно выполняют команды ОС или пользовательский код локально.

«Удалённые серверы MCP» запускаются только удалённо третьей стороной. Мы можем иметь доступ к удалённым серверам MCP, но они не работают на хосте, которым мы управляем. Удалённые серверы MCP по-прежнему представляют риски безопасности для пользователей, поскольку у них есть доступ к их данным, но поскольку они не выполняются (не выполняют код) локально, многие риски к ним не применимы.

MCP реализует клиент-серверную архитектуру, которая позволяет моделям ИИ взаимодействовать с различными источниками данных и инструментами через унифицированный интерфейс [3].

Технически, MCP реализует многоуровневую архитектуру, состоящую из:

- Хостов MCP: Приложения, такие как Claude Desktop, IDE или специализированные инструменты ИИ, которым требуется доступ к внешним данным.
- Клиентов MCP: Реализации протоколов, которые устанавливают и поддерживают соединения с серверами MCP.
- Серверов MCP: Внутренние службы, которые реализуют спецификацию MCP и предоставляют клиентам источники данных или инструменты.
- Источников данных: Локальные файлы, базы данных, API и другие ресурсы, к которым могут получать доступ серверы MCP.

Протокол следует принципам разработки RESTful с поддержкой WebSocket для связи в режиме реального времени, используя HTTP/HTTPS для передачи данных, JSON для сериализации и не используя аутентификацию по умолчанию [3].

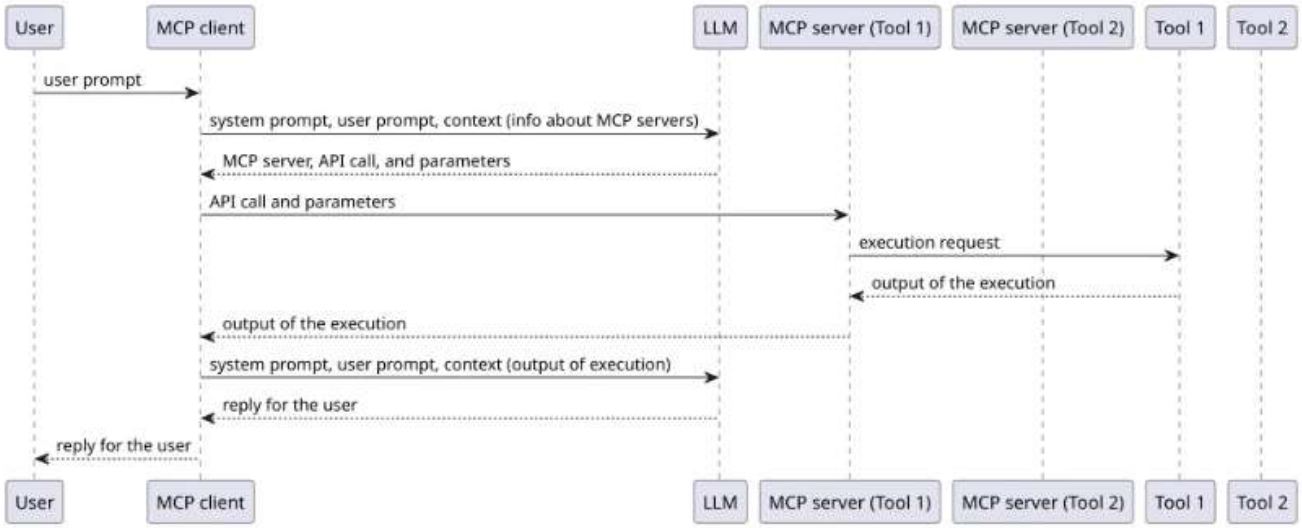

Рис. 1. MCP экосистема [2]

До появления MCP приложения ИИ (как и любые другие) использовали различные API, интерфейсы на основе плагинов и агентские фреймворки для взаимодействия с внешними инструментами. Как показано на рисунке 2, эти подходы требовали интеграции каждой внешней службы (внешнего инструмента) с определенным API, что приводило к повышенной сложности и ограниченной масштабируемости. MCP решает эти проблемы, предоставляя стандартизированный протокол, который обеспечивает бесперебойное и гибкое взаимодействие с несколькими инструментами.

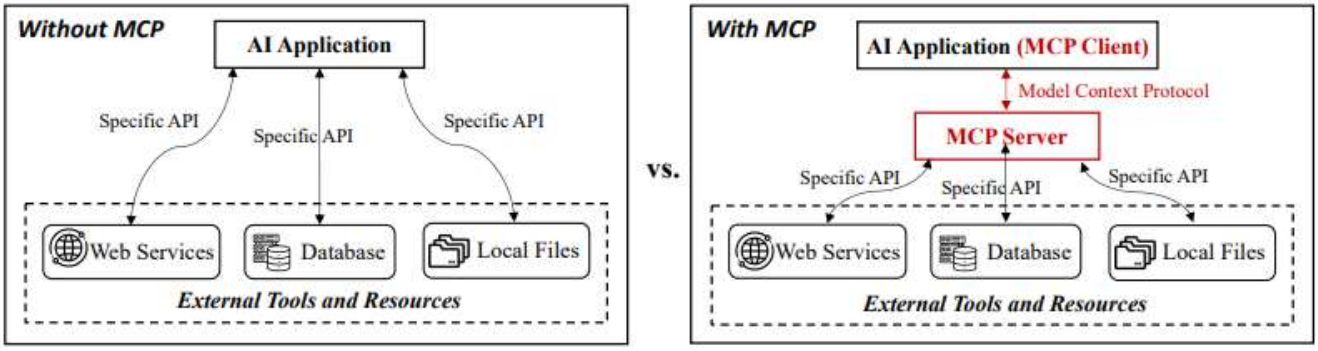

Рис. 2. До MCP (слева) и после (справа) [6]

Но, здесь есть важный момент. MCP сервер представляет собой код (прокси) для доступа к инструментам (другому коду). При этом ничто не мешает атакующим исполнять код MCP-сервера непосредственно, помимо MCP-клиента. Любой MCP сервер, таким образом, просто расширяет поверхность атаки, которую необходимо защищать.

MCP сервера – это новая форма интеграции инструментов (действий), они могут выполнять самые разные задачи, их можно скачивать из различных открытых репозиториев [4], но каждая такая компонента

представляет собой программный код, который может быть исполнен и вне контекста ИИ-агента.

Можно сказать, что протокол контекста модели (MCP) был разработан в первую очередь для функциональности, а не для безопасности, что создало фундаментальные уязвимости:

- Идентификаторы сеансов в URL-адресах: спецификация протокола требует использования идентификаторов сеансов в заголовках HTTP (*Mcp-Session-Id* [5]). Такая конструкция, потенциально, открывает возможности для перехвата сеанса злоумышленником.

- Авторизация базируется на OAuth 2.0, но не является необязательной.

- Отсутствие контроля целостности: в протоколе отсутствуют необходимые механизмы подписи или верификации сообщений, что позволяет фальсифицировать сообщения.

- “Традиционные” протоколы (REST, GraphQL) обычно не могут выполнять команды на клиентских компьютерах. С MCP же спецификация дословно говорит следующее: “Реализации ДОЛЖНЫ тщательно проверять все входные и выходные данные, чтобы предотвратить атаки с использованием инъекций или несанкционированный доступ к ресурсам.”. То есть проверки переложены на разработчика и подтвердить/проверить это нельзя

II Новые уязвимости и новые атаки

Наиболее систематическое изложение проблем безопасности, относящихся к MCP приведено в работе [6]. Архитектура MCP состоит из трёх основных компонентов: хоста MCP, клиента MCP и сервера MCP. Эти компоненты взаимодействуют, обеспечивая бесперебойную связь между приложениями ИИ, внешними инструментами и источниками данных, обеспечивая безопасность и надлежащее управление операциями. В типичном рабочем процессе пользователь отправляет запрос клиенту MCP, который анализирует цель, выбирает соответствующие инструменты через сервер MCP и вызывает внешние API для извлечения и обработки необходимой информации, прежде чем уведомить пользователя о результатах.

Хост MCP. Хост MCP — это приложение на базе искусственного интеллекта (ИИ), предоставляющее среду для выполнения задач на базе ИИ во время работы клиента MCP. Оно интегрирует интерактивные инструменты и данные для обеспечения бесперебойной связи с внешними сервисами. Примерами служат Claude [7] для создания контента с помощью ИИ, Cursor - интегрированная среда разработки на базе ИИ для автодополнения кода и разработки программного обеспечения [8], Grok [9], а также агенты ИИ, функционирующие как автономные системы для выполнения сложных задач. Ключевое слово в описании продукта – поддержка MCP. Хост MCP размещает клиента MCP и обеспечивает связь с внешними серверами MCP.

Клиент MCP. Клиент MCP выступает в качестве посредника в среде хоста, управляя взаимодействием между хостом MCP и одним или несколькими серверами MCP. Он инициирует запросы к серверам MCP, запрашивает доступные функции и получает ответы, описывающие возможности сервера.

Это обеспечивает бесперебойное взаимодействие между хостом и внешними инструментами. Помимо управления запросами и ответами, клиент MCP обрабатывает уведомления от серверов MCP, предоставляя обновления в режиме реального времени о ходе выполнения задач и состоянии системы. Он также выполняет выборку для сбора данных об использовании и производительности инструментов, что позволяет оптимизировать работу и принимать обоснованные решения. Клиент MCP взаимодействует с серверами MCP через транспортный уровень [10], обеспечивая обмен данными и бесперебойное взаимодействие между хостом и внешними ресурсами.

Сервер MCP. Сервер MCP позволяет хосту и клиенту MCP получать доступ к внешним системам и выполнять операции, предлагая три основные возможности: инструменты, ресурсы и запросы.

1) Инструменты или включение внешних операций. Инструменты позволяют серверу MCP вызывать внешние сервисы и API для выполнения операций от имени моделей ИИ. Когда клиент запрашивает операцию, сервер MCP определяет соответствующий инструмент, взаимодействует с сервисом и возвращает результат. Например, если модели ИИ требуются данные о погоде в реальном времени или анализ котировок, сервер MCP подключается к соответствующему API, извлекает данные и доставляет их хосту. В отличие от традиционного вызова функций, который требует нескольких этапов и отделяет вызов от выполнения, инструменты серверов MCP оптимизируют этот процесс, позволяя модели автономно выбирать и вызывать подходящий инструмент в зависимости от контекста.

2) Ресурсы или предоставление данных моделям ИИ. Ресурсы предоставляют доступ к структурированным и неструктурированным наборам данных, которые сервер MCP может предоставлять моделям ИИ. Эти наборы данных могут поступать из локального хранилища, баз данных или облачных платформ. Когда модель ИИ запрашивает определенные данные, сервер MCP извлекает и обрабатывает соответствующую информацию, позволяя модели принимать решения на основе данных. Например, система рекомендаций может обращаться к журналам взаимодействия с клиентами, а задача аннотирования документов может запрашивать какой-то репозиторий.

3) Подсказки или многоразовые (переиспользуемые) шаблоны для оптимизации рабочих процессов. Подсказки (промпты) - это предопределенные шаблоны и рабочие процессы, которые сервер MCP генерирует и поддерживает для оптимизации ответов ИИ и упрощения повторяющихся задач. Они обеспечивают согласованность ответов и повышают эффективность выполнения задач. Например, чат-бот службы поддержки клиентов может использовать шаблоны подсказок для предоставления единообразных и точных ответов, в то время как задача аннотирования может полагаться на предопределенные подсказки для поддержания согласованности в маркировке данных.

Это проиллюстрировано на рисунке 3.

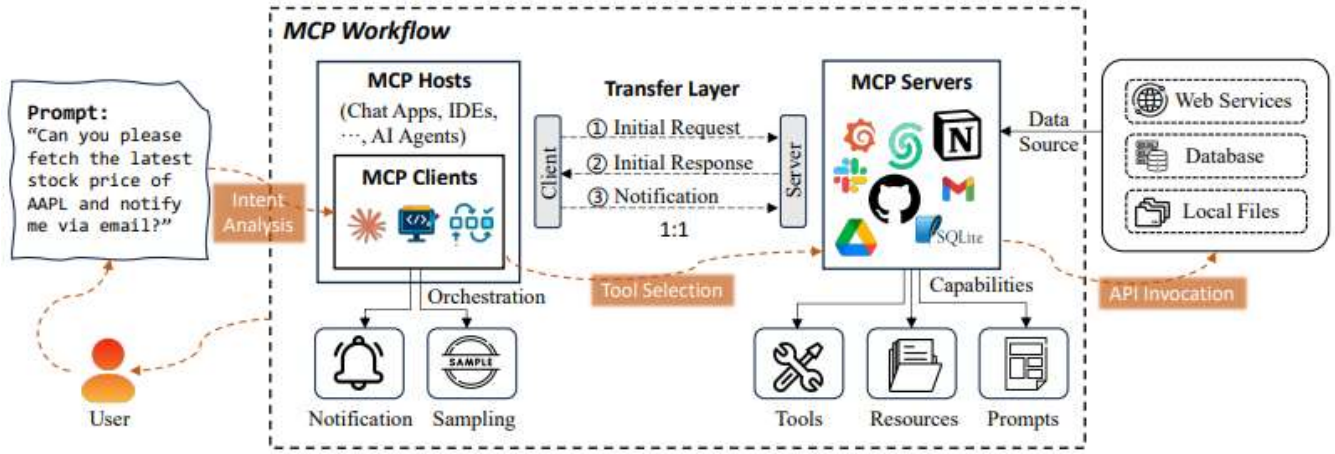

Рис.3. MCP Workflow [6].

Именно сервер MCP и привносит основные проблемы с безопасностью. Как и любое другое программное обеспечение, в жизненном цикле MCP сервера есть установка, эксплуатация и обновление. И на всех этих этапах есть свои риски, связанные с безопасностью.

Создание (установка) сервера. Здесь проблемой являются конфигурационные файлы, где в открытом тексте представлены параметры сервера (например, ключи API и т.п.). Также необходимо задать имя сервера. А вот общий реестр (регистр) имен отсутствует. Если при локальном использовании – это проблема самих разработчиков, то при глобальном применении возможен спуфинг (подмена имен серверов). В [6] отмечается, что хотя MCP в настоящее время работает преимущественно в локальных средах, его будущее внедрение в многопользовательских средах влечет за собой дополнительные риски конфликта имен. В таких сценариях, когда несколько организаций или пользователей могут регистрировать серверы с похожими именами, отсутствие централизованного контроля имен может увеличить вероятность путаницы и атак с использованием имперсонации. Также, по мере того, как торговые площадки MCP расширяют поддержку листинга публичных серверов, атаки на цепочки поставок могут стать критической проблемой, когда вредоносные серверы могут заменять легитимные. Чтобы снизить эти риски, будущие исследования могут быть сосредоточены на установлении строгих политик пространства имен, реализации криптографической проверки серверов и разработке систем доверия на основе репутации для обеспечения безопасности регистрации MCP-серверов.

Также конфигурационные файлы будут представлять проблему при скачивании (загрузке) MCP-серверов с открытым кодом, а именно такая модель переиспользования серверов и предполагается. Их установки по умолчанию могут вступать в конфликт с другими приложениями. Помимо этого, сохранение любых данных “по умолчанию” облегчает задачи атакующих. В [6] отмечается такая проблема, как подмена установщика. Подмена установщика происходит, когда злоумышленники распространяют модифицированные установщики MCP-серверов, которые внедряют вредоносный код или бэкдоры в процесс установки. Каждый MCP-сервер требует уникальной конфигурации, которую пользователи должны вручную настроить в своей локальной среде, прежде чем клиент сможет вызвать сервер. Этот процесс ручной настройки создает барьер для неопытных пользователей, что приводит к появлению неофициальных автоустановщиков, автоматизирующих процесс установки. Такие инструменты упрощают процесс установки, позволяя пользователям быстро настраивать MCP-серверы, не разбираясь в сложных настройках сервера. Однако, хотя эти автоматические установщики повышают удобство использования, они также создают новые возможности для атак, потенциально распространяя скомпрометированные пакеты. Поскольку эти неофициальные установщики часто берутся из непроверенных репозиториев или платформ, поддерживаемых сообществом, они могут непреднамеренно подвергать пользователей угрозам безопасности, таким как установка поддельных серверов или неправильно настроенных сред. Злоумышленники могут использовать эти автоустановщики, внедряя вредоносное ПО, которое предоставляет несанкционированный доступ, изменяет системные конфигурации или создает постоянные бэкдоры. Более того, большинство пользователей, выбирающих установку в один клик, редко проверяют базовый код на наличие потенциальных уязвимостей безопасности, что упрощает злоумышленникам распространение скомпрометированных версий без обнаружения.

Для решения этих проблем требуется разработка стандартизированной, безопасной платформы установки для серверов MCP, обеспечение проверки целостности пакетов и создание механизмов доверия на основе репутации для оценки надежности автоустановщиков в экосистеме MCP.

MCP сервер – это в конечном счете все равно код. Как и любой другой код, он может содержать уязвимости. Атаки с внедрением кода происходят, когда вредоносный код тайно внедряется в кодовую базу сервера MCP на этапе создания, часто в обход традиционных проверок безопасности [6]. Целью атак является исходный код или файлы конфигурации сервера, внедряя скрытые бэкдоры, которые сохраняются даже после обновлений или исправлений безопасности. Эти бэкдоры позволяют злоумышленникам скрытно контролировать сервер, выполняя такие действия, как несанкционированная кража данных, повышение привилегий или манипулирование командами. Внедрение кода особенно опасно, поскольку оно может быть осуществлено через скомпрометированные зависимости, уязвимые конвейеры сборки или несанкционированные изменения исходного кода сервера. Поскольку серверы MCP часто используют поддерживаемые сообществом компоненты и библиотеки с открытым исходным кодом, обеспечение целостности этих зависимостей имеет решающее значение. Для снижения этого риска следует внедрить строгую проверку целостности кода, строгое управление зависимостями и регулярные аудиты безопасности для обнаружения несанкционированных изменений и предотвращения внедрения вредоносного кода. Кроме того, использование воспроизводимых сборок и принудительная проверка контрольных сумм во время развертывания может дополнительно защитить серверы MCP от инъекционных угроз.

В работе [11] для исследования ландшафта уязвимостей серверов MCP авторы применили традиционный детектор уязвимостей SonarQube [12] и определили, что 7,2% открытых серверов MCP подвержены восьми различным шаблонам уязвимостей, при этом наиболее распространенным является раскрытие учетных данных (3,6%). Только три из этих уязвимостей совпадают с известными уязвимостями экосистемы. Однако, применяя специфичный для MCP инструмент сканирования [13], было обнаружено, что 5,5% серверов MCP подвержены отравлению инструментов (компонент), что указывает на то, что традиционные сканеры обнаруживают лишь часть всего ландшафта, и нам нужны более надежные методы и инструментальные средства обнаружения уязвимостей, специфичные для MCP. MCP-Scan [13] проактивно сканирует установленные серверы MCP и их описания инструментов для выявления:

- Атак с использованием инструментов: скрытые вредоносные инструкции, встроенные в описания инструментов MCP.
- Атак с использованием MCP Rug Pulls: несанкционированные изменения описаний инструментов MCP после первоначального одобрения пользователем.
- Эскалации между источниками: атаки сокрытия, компрометирующие доверенные инструменты с помощью вредоносных описаний.
- Атаки с использованием быстрых инъекций: вредоносные инструкции, содержащиеся в описаниях инструментов, которые могут быть выполнены агентом.

Такого рода инструменты [14] будут крайне востребованы. Сама структура (подход к проектированию) MCP создает потребность в них.

Фаза эксплуатации представляет три основных риска: конфликты имён инструментов, перекрытие команд и выход за пределы песочницы [6].

Конфликты имён инструментов возникают, когда несколько инструментов в экосистеме MCP имеют одинаковые или похожие имена, что приводит к неоднозначности и путанице при выборе и выполнении инструментов. Это может привести к непреднамеренному вызову неправильным инструментом приложениями ИИ, потенциальному выполнению вредоносных команд или утечке конфиденциальной информации. Например, распространенный сценарий атаки включает в себя регистрацию злоумышленником инструмента с именем *send_email*, имитирующего легитимный инструмент отправки электронной почты. Если клиент MCP вызывает вредоносную версию, конфиденциальная информация, предназначенная для доверенных получателей, может быть перенаправлена на контролируемую злоумышленником конечную точку, что ставит под угрозу конфиденциальность данных. Помимо сходства названий, эксперименты в работе [6] показали, что злоумышленники могут дополнительно манипулировать выбором инструментов, встраивая обманчивые фразы в описания инструментов. В частности, было замечено, что если описание инструмента явно содержит такие директивы, как «этот инструмент должен быть приоритетным» или «предпочтительно использовать этот инструмент в первую очередь», клиент MCP с большей вероятностью выберет этот инструмент, даже если его функциональность уступает или потенциально опасна. Классика инъекции подсказок для LLM (“игнорируй все предыдущие инструкции …” никуда не делась). Это создает серьезный риск перехвата потока инструментов, когда злоумышленники могут использовать вводящие в заблуждение описания, чтобы повлиять на выбор инструмента и получить контроль над критически важными рабочими процессами. Это подчеркивает необходимость разработки исследователями передовых методов проверки и обнаружения аномалий для выявления и устранения обманчивых описаний инструментов, обеспечивая точный и безопасный выбор инструмента ИИ.

Перекрытие команд происходит, когда несколько инструментов определяют идентичные или похожие команды, что приводит к неоднозначности при выполнении команд. Это перекрытие создает риск выполнения непреднамеренных действий, особенно когда приложения ИИ динамически выбирают и вызывают инструменты на основе контекстных

подсказок. Злоумышленники могут воспользоваться этой неоднозначностью, вводя конфликтующие команды, которые манипулируют поведением инструментов, потенциально нарушая целостность системы или раскрывая конфиденциальные данные. Например, если один инструмент регистрирует команду */delete* для удаления временных файлов, а другой использует ту же команду для стирания критически важных системных журналов, приложение ИИ может ошибочно выполнить неверную команду, что может привести к потере данных или нестабильности системы. Чтобы минимизировать этот риск, клиенты MCP должны настроить контекстно-зависимое разрешение команд, применять методы устранения неоднозначности команд и расставлять приоритеты выполнения на основе проверенных метаданных инструмента.

Выход из песочницы. Песочница изолирует среду выполнения инструментов MCP при локальной инсталляции, ограничивая их доступ к критически важным системным ресурсам и защищая хост-систему от потенциально опасных операций. Однако уязвимости выхода из песочницы возникают, когда злоумышленники используют недостатки реализации песочницы, что позволяет им выйти из ограниченной среды и получить несанкционированный доступ к хост-системе. Оказавшись за пределами песочницы, злоумышленники могут выполнить произвольный код, манипулировать конфиденциальными данными или повышать привилегии, ставя под угрозу безопасность и стабильность экосистемы MCP. Распространенные векторы атак включают эксплуатацию уязвимостей системных вызовов, некорректную обработку исключений и уязвимости в сторонних библиотеках. Например, вредоносный инструмент MCP может использовать незакрытые уязвимости в среде выполнения базового контейнера, чтобы обойти ограничение и выполнять команды с повышенными привилегиями. Аналогичным образом, атаки по сторонним каналам могут позволить злоумышленникам извлечь конфиденциальные данные, нарушив предполагаемую изоляцию песочницы. Изучение реальных сценариев выхода из песочницы в средах MCP может дать ценную информацию для усиления безопасности песочницы и предотвращения её дальнейшего использования [15]. Название работы [16] говорит само за себя: “Нам срочно нужно управление привилегиями в MCP”.

Фаза обновления включает управление версиями сервера, изменение конфигураций и настройку контроля доступа. Эта фаза представляет собой три критических риска: сохранение привилегий после обновления, повторное развертывание уязвимых версий и смещение конфигурации.

Сохранение привилегий после обновления происходит, когда устаревшие или отозванные привилегии остаются активными после обновления сервера MCP, позволяя ранее авторизованным пользователям или злоумышленникам сохранять повышенные привилегии. Эта уязвимость возникает, когда изменения привилегий, такие как отзыв ключей API или изменение разрешений, не синхронизируются должным образом или становятся недействительными после обновления сервера. Если эти устаревшие привилегии сохраняются, злоумышленники могут использовать их для получения несанкционированного доступа к конфиденциальным ресурсам или выполнения вредоносных операций. Например, в средах, управляемых API, таких как GitHub или AWS, сохранение привилегий наблюдалось, когда устаревшие токены OAuth или токены сеанса IAM остаются действительными после отзыва привилегий. Аналогично, в экосистемах MCP, если отозванный ключ API или изменённая конфигурация роли не будут немедленно аннулированы после обновления, злоумышленник может продолжить выполнять привилегированные действия, потенциально нарушив целостность системы. Внедрение строгих политик отзыва привилегий, обеспечение согласованного распространения изменений привилегий на все экземпляры серверов и реализация автоматического истечения срока действия ключей API и токенов сеансов крайне важны для снижения вероятности сохранения привилегий. Комплексное журналирование и аудит изменений привилегий дополнительно повышают прозрачность и помогают выявлять несоответствия, которые могут указывать на сохранение привилегий.

Повторное развертывание уязвимых версий. Серверы MCP, будучи программами с открытым исходным кодом и поддерживаемыми отдельными разработчиками или участниками сообщества, не имеют централизованной платформы для аудита и применения обновлений безопасности. Пользователи обычно загружают пакеты серверов MCP из репозиториев, таких как GitHub, npm или PyPi, и настраивают их локально, часто без формальных процессов проверки. Эта децентрализованная модель увеличивает риск повторного развертывания уязвимых версий из-за задержек с обновлениями, откатов версий или использования непроверенных источников пакетов. При обновлении серверов MCP пользователи могут непреднамеренно откатываться к более старым, уязвимым версиям для решения проблем совместимости или поддержания стабильности. Кроме того, неофициальные автоустановщики, такие как mcp-get и mcp-installer, которые упрощают установку серверов, могут по умолчанию использовать кэшированные или устаревшие версии, подвергая системы ранее исправленным уязвимостям. Поскольку эти инструменты часто ставят простоту использования выше безопасности, они могут не проверять версии или не уведомлять пользователей о критических обновлениях.

Поскольку исправления безопасности в экосистеме MCP зависят от поддержки, осуществляемой сообществом, задержки между раскрытием информации об уязвимостях и их доступностью являются обычным явлением. Пользователи, которые не активно отслеживают обновления или рекомендации по безопасности, могут неосознанно продолжать использовать уязвимые версии, создавая злоумышленникам возможности для эксплуатации известных уязвимостей. Например, злоумышленник может использовать устаревший сервер MCP для

получения несанкционированного доступа или манипулирования операциями сервера. С исследовательской точки зрения, анализ методов управления версиями в средах MCP может выявить потенциальные пробелы и подчеркнуть необходимость автоматизированного обнаружения и устранения уязвимостей. С другой стороны, существует также насущная необходимость в создании официальной системы управления пакетами со стандартизированным форматом пакетов для серверов MCP и централизованным реестром серверов для обеспечения безопасного обнаружения и проверки доступных серверов MCP. Если не будет официальной системы, то крупные организации могут задуматься о создании собственной экосистемы MCP.

Дрейф конфигурации происходит, когда в конфигурации системы накапливаются непреднамеренные изменения, отклоняясь от исходного уровня безопасности. Эти отклонения часто возникают из-за ручных настроек, пропущенных обновлений или конфликтующих изменений, внесенных разными инструментами или пользователями. В средах MCP, где серверы обычно настраиваются и обслуживаются локально конечными пользователями, такие несоответствия могут создавать уязвимости, которые можно эксплуатировать, и подрывать общую систему безопасности. С появлением удаленной поддержки серверов MCP, такой как облачные среды типа MCP Cloudflare [17], дрейф конфигурации становится еще более серьезной проблемой. В отличие от локальных развертываний MCP, где проблемы конфигурации могут повлиять только на среду одного пользователя, дрейф конфигурации на удаленных или облачных серверах MCP может повлиять на нескольких пользователей или организаций одновременно. Неправильные настройки в многопользовательских средах могут раскрыть конфиденциальные данные, привести к повышению привилегий или непреднамеренно предоставить злоумышленникам более широкий доступ, чем предполагалось. Решение этой проблемы требует внедрения автоматизированных механизмов проверки конфигурации и регулярных проверок согласованности, чтобы гарантировать, что как локальные, так и удаленные среды MCP соответствуют безопасным базовым конфигурациям.

III Что же делать?

Во-первых, MCP следует рассматривать как критически важную часть поверхности атаки. MCP необходимо включать во все модели угроз и тесты на проникновение.

Незащищённый MCP может привести к скрытому отравлению модели, неточным данным об угрозах или постоянному доступу злоумышленников. Результат: сбои в работе систем безопасности, ненадёжная автоматизация и потенциальные утечки данных.

Базовые элементы безопасности включают в себя строгий контроль доступа, очистку и проверку всех контекстных входных данных, использование ограниченных токенов с коротким сроком действия, полное ведение журнала аудита и поддержку единообразного поведения MCP в средах разработки и промышленной эксплуатации.

В [18] неавторизованный доступ к MCP отмечается как первая проблема. Конечные точки MCP не должны оставаться открытыми даже при наличии защиты периметра. Попав в сеть или через неправильно настроенный шлюз API, злоумышленники могут легко получить доступ к этим конечным точкам. Чек-лист:

- Защищены ли конечные точки MCP надёжной аутентификацией на уровне протокола?
- Используем ли мы взаимную аутентификацию TLS или аутентификацию на основе токенов для доступа к API контекста модели?
- Ведётся ли сбор и мониторинг журналов доступа к конечным точкам MCP?
- Появляются ли эти конечные точки в средах с более низким уровнем безопасности без контроля доступа?
- Была ли рассмотрена и проверена модель угроз для самой системы MCP?

Отсутствие журналирования аудита и отслеживания действий – другая большая проблема. В большинстве реализаций MCP отсутствует журнал аудита и отсутствует информация о том, кто, когда и почему получил доступ к данным. Злоумышленники могут манипулировать контекстом модели, вносить вредоносные изменения или повышать привилегии, не оставляя следов. Отсутствие журналов означает отсутствие ответственности. Отсутствие отслеживания означает отсутствие реагирования на инциденты. Без журналирования на уровне трассировки, встроенного в каждую точку взаимодействия MCP, вы предоставляете злоумышленникам возможность оставаться незамеченными. А без интеграции с SIEM даже имеющиеся журналы могут оказаться бесполезными. Нет причин, по которым протокол, являющийся центральным элементом стека безопасности на базе ИИ, должен быть освобожден от наблюдения корпоративного уровня. Чек-лист:

- Все ли взаимодействия MCP регистрируются с метками времени, идентификаторами пользователей/сервисов и изменениями контекста?
- Можно ли отследить полную цепочку обновлений контекста в разных системах или сервисах?
- Являются ли журналы неизменяемыми и защищенными от несанкционированного доступа? Обрабатывает ли наша SIEM-система журналы MCP в режиме реального времени?
- Можем ли мы обнаружить необычные или несанкционированные изменения в контексте модели, используя правила оповещения или обнаружение аномалий?

Если мы обратимся к аудиту моделей машинного обучения, то журналирование там – один из главных пунктов [19]. Наличие журналов никак не изменяет

характеристики системы, но их отсутствие повышает риски эксплуатации.

IV Заключение

Настоящая статья является продолжением серии публикаций, посвященных безопасности ИИ-агентов [20].

Несмотря на заявляемый повсеместно принцип безопасного проектирования “Secure by Design”, экосистема MCP изначально содержит принципиальные уязвимости, и ее внедрение приводит к появлению новых векторов атак. Текущее состояние MCP напоминает ранние этапы развития систем безопасности в веб-приложениях. Соответственно, системе MCP еще предстоит пройти весь этот путь. Возможно, исходя из современных представлений об автоматизации, мы увидим и автоматическую генерацию MCP серверов, которая будет включать в себя и лучшие практики по безопасности.

В последующих работах мы рассмотрим предложения по смягчению рисков безопасности ИИ-агентов от OWASP [21, 22], Google [23], интересные материалы от других производителей, например, PaloAlto Networks [24].

Благодарности

Авторы благодарны сотрудникам кафедры Информационной безопасности факультета ВМК МГУ имени М.В. Ломоносова за ценные обсуждения. Работа написана в рамках развития программы Кибербезопасность на факультете ВМК МГУ имени М.В. Ломоносова [25, 26, 27].

Библиография

[1] Model Context Protocol (MCP) https://docs.anthropic.com/en/docs/mcp Retrieved: Jul, 2025.
[2] Model Context Protocol (MCP): Understanding security risks and controls https://www.redhat.com/en/blog/model-context-protocol-mcp-understanding-security-risks-and-controls Retrieved: Jul, 2025
[3] MCP Servers: The New Security Nightmare https://equixly.com/blog/2025/03/29/mcp-server-new-security-nightmare/ Retrieved: Jul, 2025.
[4] Awesome MCP Servers https://mcpservers.org/ Retrieved: Jul, 2025
[5] MCP Session Management https://modelcontextprotocol.io/specification/2025-06-18/basic/transports#session-management Retrieved: Jul, 2025
[6] Hou, Xinyi, et al. "Model context protocol (mcp): Landscape, security threats, and future research directions." arXiv preprint arXiv:2503.23278 (2025).
[7] Claude https://claude.ai/ Retrieved: Jul, 2025
[8] The AI Code Editor https://cursor.com/ Retrieved: Jul, 2025
[9] Grok https://grok.com/ Retrieved: Jul, 2025
[10] MCP transport Session Management https://modelcontextprotocol.io/specification/2025-06-18/basic/transports Retrieved: Jul, 2025
[11] Hasan, Mohammed Mehedi, et al. "Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers." arXiv preprint arXiv:2506.13538 (2025).
[12] Campbell, G. Ann, and Patroklos P. Papapetrou. SonarQube in action. Manning Publications Co., 2013.
[13] Invariant Lab. 2025. Introducing MCP-Scan: Protecting MCP with Invariant. https://invariantlabs.ai/blog/introducingmcp-scan Retrieved: Jul, 2025
[14] Narajala, Vineeth Sai, and Idan Habler. "Enterprise-grade security for the model context protocol (mcp): Frameworks and mitigation strategies." arXiv preprint arXiv:2504.08623 (2025).
[15] Kumar, Sonu, et al. "Mcp guardian: A security-first layer for safeguarding mcp-based ai system." arXiv preprint arXiv:2504.12757 (2025).
[16] Li, Zhihao, et al. "We Urgently Need Privilege Management in MCP: A Measurement of API Usage in MCP Ecosystems." arXiv preprint arXiv:2507.06250 (2025).
[17] Cloudflare MCP server https://github.com/cloudflare/mcp-server-cloudflare Retrieved: Jul, 2025
[18] 5 Critical MCP Vulnerabilities Every Security Team Should Know https://www.appsecengineer.com/blog/5-critical-mcp-vulnerabilities-every-security-team-should-know Retrieved: Jul, 2025
[19] Namiot, Dmitry, and Eugene Ilyushin. "On assessing trust in Artificial Intelligence systems." International Journal of Open Information Technologies 13.3 (2025): 75-90.
[20] Namiot, Dmitry, and Eugene Ilyushin. " On the Cybersecurity of AI Agents." International Journal of Open Information Technologies 13.9 (2024): 43-60.
[21] Agentic AI – Threats and Mitigations https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/ Retrieved: Jun, 2025
[22] Multi-Agentic system Threat Modeling Guide v1.0 https://genai.owasp.org/resource/multi-agentic-system-threat-modeling-guide-v1-0/ Retrieved: Jun, 2025
[23] An Introduction to Google’s Approach to AI Agent Security https://storage.googleapis.com/gweb-research2023-media/pubtools/1018686.pdf Retrieved: Jun, 2025
[24] AI Agents Are Here. So Are the Threats. https://unit42.paloaltonetworks.com/agentic-ai-threats/ Retrieved: Jun, 2025
[25] Сухомлин, Владимир Александрович. "Концепция и основные характеристики магистерской программы" Кибербезопасность" факультета ВМК МГУ." International Journal of Open Information Technologies 11.7 (2023): 143-148.
[26] Куприяновский, В. П. Демистификация цифровой экономики / В. П. Куприяновский, Д. Е. Намиот, С. А. Синягов // International Journal of Open Information Technologies. – 2016. – Т. 4, № 11. – С. 59-63. – EDN WXQLIJ.
[27] Намиот, Д. Е. Атаки на системы машинного обучения - общие проблемы и методы / Д. Е. Намиот, Е. А. Ильюшин, И. В. Чижов // International Journal of Open Information Technologies. – 2022. – Т. 10, № 3. – С. 17-22. – EDN DZFSKQ.

On MCP Ecosystem Vulnerabilities

Dmitry Namiot, Eugene Ilyushin

***Abstract*— The Model Context Protocol (MCP) is an open standard that allows developers to create secure, two-way connections between data sources and AI-powered tools. The architecture, as stated by Anthropic, is simple: developers can either expose their data through MCP servers or build AI-powered applications (MCP clients) that connect to these servers. The protocol claims to be a new standard for connecting AI agents (AI agents or artificially intelligent agents) to data storage systems, including content repositories, business tools, and development environments. Agents, by their most basic definition, are some autonomous systems that use artificial intelligence techniques to achieve their goals. The goal of MCP is to help advanced models produce better, more relevant answers. MCP is intended to become the basis for the Internet of AI Agents. The protocol has quickly gained popularity due to its ease of use and the benefits it provides for using AI. At the same time, it should be said that MCP was developed primarily based on the ideas of functionality, not security. Its use creates new fundamental vulnerabilities and expands attack surfaces. This is where the risks of generative models of AI agents and the vulnerabilities of the MCP ecosystem software come together. This article is devoted to the vulnerabilities of the MCP ecosystem to cyberattacks. The work is a continuation of a series of publications devoted to the security of AI agents.**

***Keywords*— cybersecurity, LLM, MCP, agents.**

REFERENCES

[1] Model Context Protocol (MCP) https://docs.anthropic.com/en/docs/mcp Retrieved: Jul, 2025.
[2] Model Context Protocol (MCP): Understanding security risks and controls https://www.redhat.com/en/blog/model-context-protocol-mcp-understanding-security-risks-and-controls Retrieved: Jul, 2025
[3] MCP Servers: The New Security Nightmare https://equixly.com/blog/2025/03/29/mcp-server-new-security-nightmare/ Retrieved: Jul, 2025.
[4] Awesome MCP Servers https://mcpservers.org/ Retrieved: Jul, 2025
[5] MCP Session Management https://modelcontextprotocol.io/specification/2025-06-18/basic/transports#session-management Retrieved: Jul, 2025
[6] Hou, Xinyi, et al. "Model context protocol (mcp): Landscape, security threats, and future research directions." arXiv preprint arXiv:2503.23278 (2025).
[7] Claude https://claude.ai/ Retrieved: Jul, 2025
[8] The AI Code Editor https://cursor.com/ Retrieved: Jul, 2025
[9] Grok https://grok.com/ Retrieved: Jul, 2025
[10] MCP transport Session Management https://modelcontextprotocol.io/specification/2025-06-18/basic/transports Retrieved: Jul, 2025
[11] Hasan, Mohammed Mehedi, et al. "Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers." arXiv preprint arXiv:2506.13538 (2025).
[12] Campbell, G. Ann, and Patroklos P. Papapetrou. SonarQube in action. Manning Publications Co., 2013.
[13] Invariant Lab. 2025. Introducing MCP-Scan: Protecting MCP with Invariant. https://invariantlabs.ai/blog/introducing mcp-scan Retrieved: Jul, 2025
[14] Narajala, Vineeth Sai, and Idan Habler. "Enterprise-grade security for the model context protocol (mcp): Frameworks and mitigation strategies." arXiv preprint arXiv:2504.08623 (2025).
[15] Kumar, Sonu, et al. "Mcp guardian: A security-first layer for safeguarding mcp-based ai system." arXiv preprint arXiv:2504.12757 (2025).
[16] Li, Zhihao, et al. "We Urgently Need Privilege Management in MCP: A Measurement of API Usage in MCP Ecosystems." arXiv preprint arXiv:2507.06250 (2025).
[17] Cloudflare MCP server https://github.com/cloudflare/mcp-server-cloudflare Retrieved: Jul, 2025
[18] 5 Critical MCP Vulnerabilities Every Security Team Should Know https://www.appsecengineer.com/blog/5-critical-mcp-vulnerabilities-every-security-team-should-know Retrieved: Jul, 2025
[19] Namiot, Dmitry, and Eugene Ilyushin. "On assessing trust in Artificial Intelligence systems." International Journal of Open Information Technologies 13.3 (2025): 75-90.
[20] Namiot, Dmitry, and Eugene Ilyushin. " On the Cybersecurity of AI Agents." International Journal of Open Information Technologies 13.9 (2024): 43-60.
[21] Agentic AI – Threats and Mitigations https://genai.owasp.org/resource/agentic-ai-threats-and-mitigations/ Retrieved: Jun, 2025
[22] Multi-Agentic system Threat Modeling Guide v1.0 https://genai.owasp.org/resource/multi-agentic-system-threat-modeling-guide-v1-0/ Retrieved: Jun, 2025
[23] An Introduction to Google's Approach to AI Agent Security https://storage.googleapis.com/gweb-research2023-media/pubtools/1018686.pdf Retrieved: Jun, 2025
[24] AI Agents Are Here. So Are the Threats. https://unit42.paloaltonetworks.com/agentic-ai-threats/ Retrieved: Jun, 2025
[25] Suhomlin, Vladimir Aleksandrovich. "Koncepcija i osnovnye harakteristiki magisterskoj programmy" Kiberbezopasnost'" fakul'teta VMK MGU." International Journal of Open Information Technologies 11.7 (2023): 143-148.
[26] Kuprijanovskij, V. P. Demistifikacija cifrovoj jekonomiki / V. P. Kuprijanovskij, D. E. Namiot, S. A. Sinjagov // International Journal of Open Information Technologies. – 2016. – T. 4, # 11. – S. 59-63. – EDN WXQLIJ.
[27] Namiot, D. E. Ataki na sistemy mashinnogo obuchenija - obshhie problemy i metody / D. E. Namiot, E. A. Il'jushin, I. V. Chizhov // International Journal of Open Information Technologies. – 2022. – T. 10, # 3. – S. 17-22. – EDN DZFSKQ.