УДК 004.056:004.8:004.272

АНАЛИЗ КИБЕРАТАК НА СИСТЕМЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ЧЕРЕЗ УЯЗВИМОСТИ В ДРАЙВЕРАХ УСКОРИТЕЛЕЙ

Белкин Е.Н.

Начальник отдела научно-технического развития

АО «АЛГОНТ»,

г. Калуга, Россия

Аннотация

В статье рассмотрены киберриски систем искусственного интеллекта, возникающие на уровне драйверов и сред выполнения GPU и NPU-ускорителей. Показано, что уязвимости в драйвере, планировщике, управлении памятью, интерфейсах DMA/MMIO, механизмах профилирования и изоляции контекстов способны обходить модельные средства защиты и влиять на конфиденциальность, целостность и доступность больших языковых моделей и иных нейросетей. Обобщены теоретические предпосылки атак и академически описанные факты: побочные каналы GPU, извлечение архитектуры и параметров моделей, утечки локальной памяти, ошибки памяти в фреймворках и практичность атак на память GPU. Предложены организационные, системные и программные меры защиты. Научная новизна заключается в контекстно-ориентированной модели защитного шлюза ускорителей, связывающей состояние драйвера и оборудования с чувствительностью ИИ-актива и политикой допуска к ускорителю.

Ключевые слова: кибербезопасность, ИИ, большие языковые модели, GPU, NPU, драйвер, уязвимости, утечка памяти, состязательные атаки, изоляция контекстов.

ANALYSIS OF CYBERATTACKS ON ARTIFICIAL INTELLIGENCE SYSTEMS THROUGH VULNERABILITIES IN ACCELERATOR DRIVERS

Belkin E.N.
Head of R&D department,
ALGONT JSC,
Kaluga, Russia

Abstract

The article examines cybersecurity risks of artificial intelligence systems that arise at the level of GPU/NPU accelerator drivers and runtimes. It is shown that vulnerabilities in drivers, schedulers, memory management, DMA/MMIO interfaces, profiling mechanisms and context isolation can bypass model-level protection and affect confidentiality, integrity and availability of large language models and other neural networks. The paper summarizes theoretical attack prerequisites and academically described facts: GPU side channels, model architecture and parameter extraction, local memory leakage, memory errors in deep learning frameworks and practical attacks on GPU memory. Organizational, system and software countermeasures are proposed. The novelty is a context-oriented accelerator security gateway model that links the driver and hardware trust state with AI asset sensitivity and accelerator access policy.

Keywords: cybersecurity, AI, large language models, GPU, NPU, driver, vulnerabilities, memory leakage, adversarial attacks, context isolation.

Введение

Инфраструктура современных систем ИИ состоит не только из модели и программного фреймворка. В доверенную вычислительную базу входят драйвер ускорителя, пользовательская библиотека, компилятор графов, планировщик ядер, прошивка, интерфейсы DMA/MMIO, подсистема виртуальной памяти и средства контейнеризации. Для больших языковых моделей (БЯМ) этот уровень критичен: в памяти GPU или NPU одновременно находятся веса, LoRA-адаптеры, эмбеддинги, токены запроса, системный промпт, KV-cache, логиты и промежуточные тензоры.

Классические исследования безопасности ИИ обычно описывают атаки уклонения, отравления, извлечения модели и состязательные примеры [4; 5; 7; 11]. Эти атаки воздействуют на модельный контур. Уязвимость драйвера ускорителя относится к нижнему уровню и потому опаснее в ряде сценариев: она может дать злоумышленнику доступ к данным модели без знания архитектуры, обходить фильтрацию промптов, нарушать изоляцию контейнеров и изменять результат вывода уже после прохождения прикладных проверок.

Цель статьи - систематизировать теоретические и фактические риски атак на системы ИИ через уязвимости в драйверах GPU и NPU-ускорителей и предложить модель защиты, учитывающую чувствительность ИИ-нагрузки. Объектом исследования являются системы инференса и дообучения нейросетей, использующие аппаратные ускорители. Предмет исследования - воздействие уязвимостей драйверов и смежных компонентов ускорителя на конфиденциальность, целостность и доступность ИИ-моделей.

Драйвер ускорителя как часть доверенной базы ИИ

Драйвер GPU/NPU является привилегированным посредником между пользовательским процессом и аппаратным ускорителем. Через него выполняются создание контекстов, отображение памяти, передача командных буферов, запуск ядер, управление очередями, обработка прерываний, доступ к счетчикам производительности, сброс устройства и взаимодействие с прошивкой. В Linux и Windows значительная часть этих операций доступна непривилегированным процессам через ioctl, системные службы или runtime API. Поэтому ошибка проверки параметров, либо ошибка жизненного цикла объекта в драйвере может стать системной уязвимостью.

Для GPU характерны многопроцессный доступ, общая видеопамять, кэширование и асинхронное исполнение. Для NPU дополнительно важны локальная scratchpad-память, статическое расписание графов, NoC-коммуникации, аппаратные блоки матричных операций и специализированные компиляторы. Исследования безопасной архитектуры NPU показывают, что

защита должна охватывать не только ядра вычислений, но и каналы передачи данных, разметку памяти и среду доверенного исполнения [12; 21; 23].

Отличие от обычного серверного приложения состоит в том, что в памяти ускорителя концентрируется высокоценный ИИ-актив. Для БЯМ это не только веса модели. В процессе инференса появляются временные данные, по которым можно восстановить фрагменты запроса или ответа, выявить служебные инструкции, получить признаки домена и пользователя. Утечка такой памяти может быть эквивалентна утечке журнала запросов, закрытых документов или корпоративной базы знаний.

Модель угроз

Рассматривается локальный или контейнерный нарушитель, имеющий возможность запускать вычисления на том же ускорителе, что и целевая ИИ-нагрузка, но не имеющий административных прав на модельный сервис. Это типовой сценарий общего Jupyter/Kubernetes-кластера, облачной среды с совместным использованием GPU, лабораторного сервера, edge-устройства или рабочей станции разработчика. Более сильный нарушитель может контролировать один контейнер инференса или подменить плагин фреймворка.

Цели нарушителя: 1) чтение промптов, выходных данных, эмбеддингов, KV-cache, LoRA-адаптеров и весов; 2) изменение промежуточных тензоров, весов или логитов для искажения ответа; 3) отказ в обслуживании через сброс драйвера, перегрузку памяти или зависание очередей; 4) повышение привилегий через ошибку драйвера и переход к компрометации узла; 5) извлечение архитектурных характеристик модели для дальнейших атак.

В таблице 1 приведена обобщенная классификация атак. Она объединяет результаты работ по атакам на модели машинного обучения [1; 2; 4; 5; 7; 8; 10; 11] и публикации по аппаратно-программным каналам ускорителей [12-24].

Таблица 1 - Классы атак через драйверы и среду выполнения ускорителей

Класс воздействия	Участок ускорителя	Риск для ИИ-системы

Нарушение изоляции памяти	Выделение и освобождение VRAM/SRAM, локальная память, кэши, mmap	Чтение промптов, ответов, KV-cache, эмбеддингов, весов, LoRA-адаптеров; утечка межконтейнерных данных
Побочный канал	Счетчики производительности, время исполнения, конкуренция за кэши и память, контекстные переключения	Извлечение архитектуры модели, размеров слоев, паттернов запросов и признаков приватных данных
Повреждение данных	Ошибки драйвера, некорректная адресация, гонки, ошибки ядра ускорителя, bit-flip в памяти	Искажение логитов, скрытая деградация качества, целевая ошибка классификации, нарушение политики генерации
Компрометация драйвера	ioctl, командные буферы, DMA/MMIO, прошивка, компилятор графов	Повышение привилегий, обход контейнера, доступ к секретам хоста и к моделям других пользователей
Отказ в обслуживании	Планировщик, watchdog, сброс устройства, память, очередь команд	Остановка инференса, потеря SLA, разрыв цепочки автоматизированного принятия решений

Известные факты уязвимостей и атак

Академическая литература подтверждает, что аппаратно-программный уровень ускорителей не является только теоретическим источником риска. Работа Rendered Insecure показала практичность побочных каналов GPU через API выделения памяти, счетчики производительности и временные измерения [19]. Leaky DNN описывает извлечение секретов модели с использованием побочного канала контекстного переключения GPU [24]. DeepSniffer демонстрирует возможность извлечения архитектуры DNN по подсказкам, получаемым при исполнении на GPU [13].

Особенно значим для БЯМ факт, описанный в LeftoverLocals: остаточные данные локальной памяти GPU могут раскрывать ответы LLM между процессами или контейнерами [22]. Это показывает, что даже без классической prompt-инъекции нарушитель, соседствующий на ускорителе, может получить

данные диалога. Подобный риск не устраняется фильтрами входных сообщений, поскольку возникает ниже уровня токенизатора и модели.

Исследования GPUHammer показывают практичность атак на память GPU класса Rowhammer и возможность деградации точности моделей при инжекции ошибок в память [17]. Это важно для ИИ-инференса, так как единичные ошибки в весах, таблицах эмбеддингов или промежуточных тензорах могут не вызвать сбой процесса, но изменить распределение вероятностей токенов или результат классификации. Работы GPU-Fuzz и CuSafe подтверждают актуальность ошибок памяти в глубоком обучении и необходимость специализированных средств обнаружения повреждений на GPU [16; 18].

Для NPU опубликованы отдельные направления защиты и атак. NPUFort рассматривает архитектуру защиты DNN-ускорителя от инверсии модели [23]. sNPU предлагает доверенную среду исполнения для интегрированных NPU, включая изоляцию памяти и каналов [12]. NeuroPlug направлен на снижение утечек побочных каналов в NPU [21]. Эти работы показывают, что нейропроцессоры имеют собственные поверхности атаки: локальные буферы, статическое расписание графа, специализированные межсоединения и компиляторные преобразования.

Обобщение известных фактов приведено в таблице 2. В таблицу включены только академические публикации; сообщения вендоров и базы CVE учитываются как практический фон, но не используются как источники библиографического списка.

Таблица 2 - Академически описанные факты, значимые для БЯМ и других ИИ-моделей

Источник	Факт	Значение для систем ИИ
[19]	Практические побочные каналы GPU через память, счетчики производительности и время.	Возможность вывода признаков чужой нагрузки и поведения модели при совместном использовании ускорителя.

[24]	Побочный канал GPU context-switching для похищения секретов DNN.	Риск восстановления архитектурных параметров и режимов исполнения модели.
[13]	Извлечение архитектуры DNN по аппаратно-программным подсказкам исполнения.	Упрощение атак белого ящика, подбора состязательных входов и копирования модели.
[22]	Утечки локальной памяти GPU позволяют слушать ответы LLM из другого процесса или контейнера.	Прямая угроза промптам, ответам, системным инструкциям и данным RAG-контекста.
[17]	Rowhammer на памяти GPU может вызывать bit-flip и снижать точность ML-моделей.	Нарушение целостности весов или тензоров без явного отказа сервиса.
[12; 23]	Для NPU требуются TEE, изоляция scratchpad/NoC и защита от инверсии.	NPU нельзя считать автоматически безопаснее GPU; защита должна быть архитектурной.
[14; 16; 18]	Фаззинг и санитайзинг выявляют ошибки драйверов и памяти в GPU-стеке.	Необходима регулярная проверка драйверов, runtime, ядер и ML-фреймворков.

Риски влияния на работу БЯМ и иных моделей

Конфиденциальность: В БЯМ пользовательский запрос и ответ проходят через токенизатор, embedding lookup, attention-блоки, KV-cache и слои генерации. При пакетном инференсе в памяти одного ускорителя находятся фрагменты нескольких запросов. Если память не очищается при освобождении или при смене контекста, нарушитель может прочитать остаточные данные. Наибольший риск имеют корпоративные RAG-системы, ассистенты с доступом к документам, медицинские и юридические помощники, а также сервисы программирования, где в промптах могут содержаться ключи, фрагменты исходного кода и внутренние архитектурные сведения.

Целостность: Уязвимость драйвера или ошибка памяти может привести к несанкционированной записи в буфер весов, активаций или логитов. Последствия зависят от участка. Ошибка в embedding-таблице меняет смысл токенов; ошибка в KV-cache искажает контекст; ошибка в logits processor может менять распределение вероятностей; повреждение LoRA-адаптера способно вводить доменную закладку. Для классификаторов и систем компьютерного

зрения результатом становится неверный класс; для БЯМ - ложный вывод, обход ограничений или отказ отвечать на допустимые запросы.

Доступность: GPU/NPU-драйверы обслуживают дорогостоящий общий ресурс. Некорректный командный буфер, зависшее ядро, исчерпание VRAM или ошибка планировщика вызывают сброс устройства и остановку всех процессов на ускорителе. В ИИ-сервисе это приводит к массовым тайм-аутам, потере очереди запросов и ошибкам в зависимых бизнес-процессах.

Привилегии и горизонтальное распространение. Ускорители часто пробрасываются в контейнеры как устройства хоста. Если контейнер получает избыточные права или доступ к служебным файлам драйвера, уязвимость может использоваться для выхода за границы контейнера. Поэтому безопасность ИИ-кластера нельзя оценивать только по настройкам API-сервиса; нужно проверять путь от пользователя до /dev/nvidia*, /dev/dri, runtime-библиотек и системных daemon.

Репутационные и правовые последствия: Утечка промптов или ответов может раскрыть персональные данные, коммерческую тайну и сведения о внутренних процессах. Скрытая деградация модели опасна тем, что не всегда выглядит как инцидент информационной безопасности: сервис продолжает отвечать, но качество и безопасность вывода ухудшаются.

Методы противодействия

Меры защиты должны сочетать управление уязвимостями драйвера, изоляцию сред выполнения и контроль собственно ИИ-модели. Недостаточно обучить модель на состязательных примерах [3; 10; 11], если соседний процесс способен читать остаточную память ускорителя или вызвать сброс драйвера. Также недостаточно регулярно обновлять драйвер, если модельные веса и KV-cache не классифицируются как защищаемые активы.

Базовые практики: инвентаризация версий драйвера, firmware, runtime и ML-фреймворков; утвержденная матрица совместимости; запрет неподдерживаемых версий; оперативное применение исправлений;

тестирование обновлений на эталонных ИИ-нагрузках; контроль цифровой подписи драйверов и прошивок; secure boot, measured boot и проверка целостности модулей ядра.

Изоляция: запрет запуска ИИ-контейнеров в privileged-режиме; минимальный набор устройств; запрет произвольного hostPath к каталогам драйвера; seccomp/AppArmor/SELinux-профили; отдельные пулы ускорителей для разных уровней доверия; IOMMU; отключение профилировщиков и счетчиков производительности для непривилегированных пользователей; запрет совместного использования ускорителя для задач с высокой чувствительностью. MIG, SR-IOV и vGPU следует считать средствами уменьшения риска, но не полным барьером без подтвержденной модели угроз.

Гигиена памяти: очистка VRAM/SRAM при освобождении буфера, завершении процесса и смене арендатора; сброс устройства перед передачей высокочувствительной нагрузки другому пользователю; контроль фреймворков на предмет повторного использования буферов; отдельное хранение системных промптов и секретов; минимизация длины KV-cache для чувствительных задач; исключение попадания ключей доступа в промпты; шифрование данных на диске и защита каналов передачи до узла ускорителя.

Контроль целостности ИИ: подпись и хэширование весов, токенизаторов, LoRA-адаптеров и графов компиляции; проверка хэшей перед загрузкой и по расписанию; эталонные canary-запросы; мониторинг дрейфа распределения логитов и качества; резервный инференс на другом ускорителе для критических решений; включение ECC там, где это поддерживается; регистрация ошибок ECC, Xid/driver reset и отказов ядра.

Тестирование: фаззинг ioctl и пользовательских библиотек, проверка ML-компиляторов и операторов, динамический анализ GPU-ядер, тесты на остаточную память и гонки, негативные тесты выделения/освобождения буферов. Работы Moneta, GPU-Fuzz и CuSafe показывают целесообразность специализированных инструментов для GPU-стека, так как обычные

санитайзеры CPU не покрывают асинхронную модель памяти ускорителей [14; 16; 18].

Мониторинг: сбор событий драйвера, аномальных сбросов, обращений к профилировщикам, частоты контекстных переключений, ошибок памяти, длительных ядер, нетипичных паттернов выделения VRAM, признаков конкурентного давления на память и отклонений модели. Важна корреляция системной телеметрии с метриками ИИ-сервиса: токены в секунду, доля отказов, длина ответа, частота галлюцинаций, изменение профиля логитов.

Контекстно-ориентированная модель защитного шлюза ускорителей

В качестве научной новизны предлагается контекстно-ориентированная модель защитного шлюза ускорителей (КОШУ). Ее отличие от обычного hardening-подхода состоит в том, что решение о допуске к GPU/NPU принимается не только по версии драйвера и правам пользователя, но и по чувствительности конкретной ИИ-нагрузки. Модель связывает системную безопасность с семантикой ИИ-актива.

КОШУ включает семь функций. 1) Сенсор доверенного состояния: версии драйвера, firmware, runtime, ML-фреймворка, признаки secure boot, IOMMU, ECC, доступность профилировщиков, события сброса устройства. 2) Классификатор ИИ-актива: публичная модель, внутренняя модель, модель с конфиденциальным RAG-контекстом, критическая модель принятия решений. 3) Брокер политики доступа: выбор режима shared, partitioned, dedicated или TEE. 4) Контроллер памяти: очистка буферов на alloc/free, завершении процесса и смене арендатора. 5) Контроллер целостности: подписи, хэши, canary-инференс, контроль адаптеров. 6) Защита от побочных каналов: отключение счетчиков, квоты на профилирование, раздельное планирование несовместимых арендаторов. 7) Коррелятор аномалий: сопоставление телеметрии драйвера с метриками качества модели.

Риск допуска к ускорителю вычисляется как $R = S \times (Vd + Vi + M + T) \times E$, где S - чувствительность ИИ-актива от 1 до 5; Vd - оценка уязвимости драйвера

и runtime; Vi - недостаточность изоляции; M - риск остаточной памяти; T - текущая аномалия телеметрии; E - коэффициент экспозиции среды. Для одиночного доверенного узла E = 0,5; для общего внутреннего кластера E = 1; для публичной многоарендной среды E = 1,5.

Политика КОШУ: при R < 10 допускается общий режим с обязательной очисткой памяти; при 10 <= R < 25 требуется разделение арендаторов, запрет профилировщиков и контроль хэшей; при R >= 25 требуется выделенный ускоритель или доверенная среда NPU/GPU, полный сброс между задачами, canary-проверка и резервный инференс для критических решений. При обнаружении события сброса драйвера, ошибки ECC высокой кратности или несоответствия хэша модель переводится в режим карантина.

Практическая реализация возможна как расширение device plugin для Kubernetes, модуль admission control и локальный агент на узле. Агент не должен перехватывать содержимое промптов; ему достаточно меток чувствительности, системной телеметрии и контрольных хэшей. Это уменьшает риск дополнительной утечки данных и позволяет применять модель в закрытых контурах.

Ожидаемый эффект КОШУ: снижение вероятности межарендной утечки, раннее обнаружение атак на память и драйвер, формальная привязка политики изоляции к ценности ИИ-актива, возможность аудита решений о размещении модели. Модель совместима с традиционными мерами защиты от состязательных атак на уровне входных данных, но закрывает иной слой угроз - аппаратно-программный контур ускорителя.

Выводы

Уязвимости драйверов GPU/NPU и смежных runtime-компонентов создают отдельный класс рисков для систем ИИ. Они воздействуют ниже уровня промпта, токенизатора и модельных фильтров, поэтому могут нарушать конфиденциальность, целостность и доступность даже у моделей, устойчивых к традиционным состязательным входам.

Для БЯМ наиболее опасны остаточная память, побочные каналы, ошибки изоляции контекстов, некорректная обработка командных буферов, повреждение памяти и отказ драйвера. Известные академические работы подтверждают практичность таких рисков для GPU и необходимость специальных архитектурных решений для NPU.

Защита должна строиться как многоуровневая система: hardening драйвера и ОС, строгая контейнерная изоляция, гигиена памяти ускорителя, контроль целостности моделей, специализированное тестирование GPU/NPU-стека и корреляция системной телеметрии с метриками ИИ. Предложенная модель КОШУ добавляет управляемую политику допуска к ускорителю на основе чувствительности ИИ-актива и текущего состояния доверенной базы.

Библиографический список:

1. Биджиев Т.М., Намиот Д.Е. Исследование существующих подходов к встраиванию вредоносного программного обеспечения в искусственные нейронные сети // International Journal of Open Information Technologies. - 2022. - Т. 10. - № 9. - С. 21-31.

2. Володин И.В., Путято М.М., Макарян А.С., Евглевский В.Ю. Классификация механизмов атак и исследование методов защиты систем с использованием алгоритмов машинного обучения и искусственного интеллекта // Прикаспийский журнал: управление и высокие технологии. - 2021. - № 2 (54). - С. 91-98.

3. Герасимов В.М., Маслова М.А., Халилаева Э.И. Защита от состязательных атак на аудио и изображения в моделях искусственного интеллекта с применением метода SGEC // Научный результат. Информационные технологии. - 2023. - Т. 8. - № 2. - С. 53-60.

4. Ильюшин Е.А., Намиот Д.Е., Чижов И.В. Атаки на системы машинного обучения - общие проблемы и методы // International Journal of Open Information Technologies. - 2022. - Т. 10. - № 3. - С. 17-22.

5. Котенко И.В., Саенко И.Б., Лаута О.С., Васильев Н.А., Садовников В.Е. Атаки и методы защиты в системах машинного обучения: анализ современных исследований // Вопросы кибербезопасности. - 2024. - № 1. - С. 24-37.

6. Лапина М.А., Ржевская Н.В., Котляров Д.В., Дюдюн Г.Д. Особенности организации атак на нейронные сети для распознавания образов // Auditorium. - 2023. - № 2 (38). - С. 97-103.

7. Намиот Д.Е. Схемы атак на модели машинного обучения // International Journal of Open Information Technologies. - 2023. - Т. 11. - № 5. - С. 68-86.

8. Намиот Д.Е., Ильюшин Е.А., Чижов И.В. Искусственный интеллект и кибербезопасность // International Journal of Open Information Technologies. - 2022. - Т. 10. - № 9. - С. 135-147. - DOI: 10.5281/zenodo.12573114.

9. Сивков Д.И., Федосенко М.Ю. Атаки и методы защиты при использовании методов машинного обучения в контексте стегоанализа цифрового контента // Экономика и качество систем связи. - 2024. - № 3. - С. 136-145.

10. Тищенко Д.А., Приходько Т.А. Алгоритм защиты от состязательных атак в языковых моделях // International Journal of Open Information Technologies. - 2024. - Т. 12. - № 10. - С. 34-39.

11. Фомичева С.Г., Беззатеев С.В. Механизмы защиты моделей машинного обучения от состязательных атак // T-Comm: Телекоммуникации и транспорт. - 2023. - Т. 17. - № 10. - С. 28-42. - DOI: 10.36724/2072-8735-2023-17-10-28-42.

12. Feng E., Feng D., Du D., Xia Y., Chen H. sNPU: Trusted Execution Environments on Integrated NPUs // Proceedings of the 51st Annual International Symposium on Computer Architecture. - 2024. - P. 708-723. - DOI: 10.1109/ISCA59077.2024.00057.

13. Hu X. et al. DeepSniffer: A DNN Model Extraction Framework Based on Learning Architectural Hints // Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems. - 2020. - DOI: 10.1145/3373376.3378460.

14. Jung J. et al. Moneta: Ex-Vivo GPU Driver Fuzzing by Recalling In-Vivo Execution States // Proceedings of the Network and Distributed System Security Symposium. - 2025.
15. Lee K. et al. Secure Machine Learning Hardware: Challenges and Progress // IEEE Circuits and Systems Magazine. - 2025. - Vol. 25. - No. 1. - P. 8-34.
16. Li Z. et al. GPU-Fuzz: Finding Memory Errors in Deep Learning Frameworks // arXiv preprint arXiv:2602.10478. - 2026.
17. Lin C.S., Qu J., Saileshwar G. GPUHammer: Rowhammer Attacks on GPU Memories are Practical // Proceedings of the USENIX Security Symposium. - 2025.
18. Lu H. et al. CuSafe: Capturing Memory Corruption on NVIDIA GPUs // Proceedings of the USENIX Security Symposium. - 2026.
19. Naghibijouybari H., Neupane A., Qian Z., Abu-Ghazaleh N.B. Rendered Insecure: GPU Side Channel Attacks are Practical // Proceedings of the ACM SIGSAC Conference on Computer and Communications Security. - 2018. - P. 2139-2153. - DOI: 10.1145/3243734.3243831.
20. Nayan T. et al. All You Need to Know About On-Device ML Model Extraction // Proceedings of the USENIX Security Symposium. - 2024.
21. Shrivastava N., Sarangi S.R. NeuroPlug: Plugging Side-Channel Leaks in NPUs using Space Filling Curves // arXiv preprint arXiv:2407.13383. - 2024.
22. Sorensen T., Khlaaf H. LeftoverLocals: Listening to LLM Responses Through Leaked GPU Local Memory // arXiv preprint arXiv:2401.16603. - 2024.
23. Wang X., Hou R., Zhu Y., Zhang J., Meng D. NPUFort: A Secure Architecture of DNN Accelerator Against Model Inversion Attack // Proceedings of the 16th ACM International Conference on Computing Frontiers. - 2019. - P. 190-196. - DOI: 10.1145/3310273.3323070.
24. Wei J. et al. Leaky DNN: Stealing Deep-Learning Model Secret with GPU Context-Switching Side-Channel // Proceedings of the 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. - 2020. - P. 125-137. - DOI: 10.1109/DSN48063.2020.00031