ROCK'N'BLOCK

Аудит смарт-контрактов

Проект: GLX Core — Gold Token Suite (GLX · Gold Bar NFT · Gold Vault)

Скоуп: GLXTokenUpgradeable.sol · GoldBarNFTUpgradeable.sol · GoldVaultUpgradeable.sol · инструментарий управления обновлениями

Архитектура: OZ v4 TransparentUpgradeableProxy + TimelockController

Целевая сеть: EVM (Cancun-совместимая)

Язык: Solidity ^0.8.23

Аудитор: Rock'n'Block

Дата: 25.06.2026

Версия отчёта: 0.2 (draft, после раунда исправлений)

Статус: CONFIDENTIAL

Версия 0.2 учитывает раунд исправлений команды разработки по итогам первичного аудита (v0.1) и фиксирует статус по каждой находке. Все замечания уровня MEDIUM устранены (5 — изменениями в коде/конфигурации, 1 — осознанно принято с операционными обязательствами); замечания уровня LOW/INFO исправлены либо приняты. Открытых пунктов не осталось. Корректность исправлений подтверждена локальным прогоном тестов (Foundry-инварианты и регрессионный набор Hardhat — без отказов).

1. Резюме

Проведён ручной аудит трёх обновляемых (upgradeable) контрактов GLX Core, а также инструментария управления обновлениями и пакетов транзакций развёртывания (Safe). Системный инвариант — `totalSupply(GLX) == GoldVault.backingFineWeight()` (с учётом `cancelledLiability()` при юридически обусловленном списании). Контракты характеризуются высокой степенью зрелости и подробной спецификацией; критических дефектов логики не обнаружено.

Сводка по severity

Severity	Количество	Комментарий
CRITICAL	0	—
HIGH	0	—
MEDIUM	6	Конфигурация развёртывания и инструментария; централизация
LOW	14	Нежелательные паттерны, превентивное усиление, операционные риски
INFO	8 + сводные	Стиль кода, оптимизация газа, проектные допущения, документация

Сводка по статусам (после раунда исправлений)

Статус	Количество	Значение
Fixed	15	Устранено изменениями в коде или конфигурации
Acknowledged	13	Принято осознанно; компенсируется операционно / by-design
Open	0	Нерешённых замечаний не осталось

Распределение по объектам

Объект	MEDIUM	LOW	INFO
GLXTokenUpgradeable.sol	1	3	3
GoldBarNFTUpgradeable.sol	0	4	1
GoldVaultUpgradeable.sol	1	3	2
Инструментарий обновлений и пакеты развёртывания	4	3	2
Репозиторий	0	1	—

Общая оценка

Архитектура механизмов контроля является зрелой: принудительная заморозка регуляторного эскроу (постоянный инвариант), правила защищённых аккаунтов (VAULT_ROLE / эскроу / Liquidity Safe), защитные проверки учёта (`EscrowDrained` / `EscrowDirty`), обязательная проверка `maxSupply` на обоих путях эмиссии, защита от повторного входа в операциях compliance, управление обновлениями через TimelockController. По итогам раунда исправлений открытых замечаний не осталось.

Подтверждение тестированием (исправленная версия)

- **Инварианты Foundry** (FOUNDRY_PROFILE=glx): пройдены 4 из 4, 4096 вызовов на инвариант, 0 откатов. Изменение раскладки хранилища регрессий не вызвало.
- **Регрессионный набор Hardhat**: 246 тестов пройдено, 0 отказов (полный набор из 11 файлов, включая тест генератора timelock-пакетов; новый набор проверок комиссии ликвидности и топологии обновлений).

2. Скоуп аудита

Контракты в скоупе

- src/contracts/GLXTokenUpgradeable.sol — ERC-20 GOLDEX/GLX, 18 dec, compliance-слой.
- src/contracts/GoldBarNFTUpgradeable.sol — ERC-721 GBAR, реестр слитков + lifecycle.
- src/contracts/GoldVaultUpgradeable.sol — backing-пул, supply-контроллер, redemption.

Инструментарий и конфигурация

- src/ts/ — validate-upgrade, генераторы пакетов Safe/Timelock, предсказатель CREATE2, verify_glx_role_topology.
- src/config/safe/*.json — пакеты транзакций Safe (развёртывание, обновление, TimelockController, назначение ролей); конвейер CI.

Вне скоупа

Исходный код наследия CoW Protocol / Uniswap / Stablecoin; необновляемые исторические варианты контрактов; внеблокчейн-процессы (KYC/AML, хранение активов, Proof-of-Reserves).

3. Методология

1. **Ручной анализ кода** — построчная проверка логики, инвариантов и отклонений от спецификации.
2. **Моделирование угроз** — модель угроз для каждой привилегированной роли и сценарии компрометации.
3. **Анализ архитектуры** — механизм обновлений (ERC-1967 + Transparent Proxy + TimelockController), раскладка хранилища, жизненный цикл, взаимодействие контрактов.
4. **Соответствие стандартам** — EIP-2612/712, ERC-165, ERC-721, ERC-7201, Solidity Style Guide.
5. **Динамическая верификация** — локальный запуск регрессионных наборов Hardhat и инвариантов Foundry (фаззинг-тестирование).

Статусы

Fixed — устранено в коде/конфигурации; **Acknowledged** — принято осознанно (компенсируется операционно либо предусмотрено архитектурой); **Open** — не устранено. В настоящей версии открытых замечаний нет.

4. Сводный реестр находок

MEDIUM

ID	Объект	Замечание	Статус
GLX-M-01	GLXToken	Централизация: роль COMPLIANCE_ROLE может заморозить и изъять GLX любого держателя	**Acknowledged**
VLT-M-01	GoldVault	Пакет назначения ролей не выдаёт LIQUIDITY_SAFE_ROLE — риск блокировки депозитов	**Fixed**
TL-M-01	Инструментарий	Генератор пакетов Timelock не проверяет равенство администратора прокси целевому TimelockController	**Fixed**
TL-M-02	Инструментарий	verify_glx_role_topology не проверяет инвариант «администратор прокси = TimelockController»	**Fixed**
TL-M-03	Инструментарий	validate-upgrade.ts работает по принципу fail-open: пропускает цели/эталоны, сообщая об успехе	**Fixed**
CFG-M-01	Конфиг	Конфигурация верификатора не содержит LIQUIDITY_SAFE_ROLE — пропуск гранта не выявляется	**Fixed**

LOW

ID	Объект	Замечание	Статус
GLX-L-01	GLXToken	seizeVaultEscrowToRegulatoryEscrow / setRegulatoryEscrow без проверки защищённых аккаунтов	**Fixed**
GLX-L-02	GLXToken	burn(address from, uint256) — избыточный параметр from	**Fixed**
GLX-L-03	GLXToken	rescueERC20 запрещает вывод GLX с адреса самого токена	**Acknowledged**
GBAR-L-01	GoldBarNFT	Отзыв одобрения склада блокирует уже выпущенный слиток до его депонирования	**Acknowledged**
GBAR-L-02	GoldBarNFT	markDeliveredAndCancel для несуществующего токена возвращает типовую ошибку OpenZeppelin	**Fixed**
GBAR-L-03	GoldBarNFT	Корректность releaseTo опирается на инвариант «vault не может быть заморожен»	**Acknowledged**
GBAR-L-04	GoldBarNFT	_beforeTokenTransfer не проверяет batchSize==1	**Fixed**
VLT-L-01	GoldVault	setLiquiditySafe не отклоняет заморожённый адрес или адрес без требуемой роли	**Fixed**
VLT-L-02	GoldVault	Поле redemptionLocked в DepositRecord нигде не читается (неиспользуемое хранилище)	**Fixed**
VLT-L-03	GoldVault	_eligibleIndexOf хранит uint256, тогда как достаточно булевой семантики	**Fixed**
TL-L-01	Инструментарий	Эталоны хранилища являются снимком текущих контрактов	**Acknowledged**
TL-L-02	Инструментарий	Расхождение обработки соли CREATE2: предсказатель дополняет нулями, генератор отклоняет	**Fixed**
CFG-L-01	Конфиг	Шаблон обновления с миграцией: эталон именуется по контракту V2, но должен содержать раскладку V1	**Acknowledged**

INFORMATIONAL

ID	Объект	Замечание	Статус
GLX-I-01	GLXToken	Соответствие GoldVault.liquiditySafe и LIQUIDITY_SAFE_ROLE не обеспечивается на уровне контракта	**Fixed**
GLX-I-02	GLXToken	seizeFrozenToEscrow генерирует событие FrozenFundsSeized при amount==0	**Acknowledged**
GLX-I-03	GLXToken	Переопределение _approve блокирует approve/permit при паузе	**Acknowledged**
GBAR-I-01	GoldBarNFT	BarMetadata хранит параметры, принудительно равные константам	**Acknowledged**
VLT-I-01	GoldVault	Комиссия ликвидности: для погашения требуется полный fineWeight (предусмотрено архитектурой)	**Acknowledged**
VLT-I-02	GoldVault	Допущение однородности слитков: FIFO-выбор корректен только при едином профиле	**Acknowledged**
TL-I-01	Инструментарий	Режим createMode:"create" не регистрирует алиасы (предусмотрено: используется только create2)	**Acknowledged**
TL-I-02	Инструментарий	Верификатор сопоставляет роли по именам необновляемых артефактов	**Acknowledged**
***-I-св.**	Все	Стиль и оптимизация газа (сводно): дублирование в seize*, префиксный инкремент, интерфейсы, порядок инициализации, .gitignore	**Частично**

5. Детальные находки

Формат описания каждого замечания: метаданные → описание → воздействие и вероятность → рекомендация → решение команды (по итогам раунда исправлений).

5.1 GLXTokenUpgradeable.sol

GLX-M-01

Централизация: роль COMPLIANCE_ROLE может заморозить и изъять GLX любого держателя

MEDIUM | GLXTokenUpgradeable.sol:220-318 | Статус: **Acknowledged**

Роль COMPLIANCE_ROLE позволяет заморозить произвольный адрес и затем изъять весь или часть его баланса GLX в пользу regulatoryEscrow с использованием механизма обхода для compliance-операций (флаг `_complianceOperationInProgress` отключает проверки заморозки и паузы). Это предоставляет роли возможность принудительно блокировать и перемещать средства любого пользователя (за исключением защищённых аккаунтов VAULT_ROLE, regulatoryEscrow и Liquidity Safe).

Указанное поведение соответствует осознанной модели регулируемого актива (по аналогии с USDC/USDT) и задокументировано в docs/glx-token.md. Тем не менее оно представляет собой существенное допущение о доверии: компрометация мультиподписного кошелька COMPLIANCE приводит к массовой заморозке и изъятию средств.

Воздействие: высокое для затронутых балансов. **Вероятность:** требует компрометации или недобросовестных действий мультиподписного кошелька COMPLIANCE.

РЕКОМЕНДАЦИЯ

Зафиксировать как принятое допущение о доверии (Acknowledged) и раскрыть в разделе рисков: роль COMPLIANCE_ROLE должна быть закреплена за отдельным проверенным мультиподписным кошельком с пороговой схемой подписей, отличным от кошельков admin/redemption/delivery; обеспечить мониторинг событий AddressFrozen и FrozenFundsSeized; рассмотреть введение временной задержки или двухэтапного подтверждения для несрочных массовых операций изъятия.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: модель доверия задокументирована (раздел trust model в API-доке GLXToken); операционное обязательство — отдельный мультиподписной кошелёк COMPLIANCE с пороговой схемой и мониторинг событий заморозки/изъятия.

GLX-L-01

seizeVaultEscrowToRegulatoryEscrow / setRegulatoryEscrow без проверки защищённых аккаунтов

LOW | GLXTokenUpgradeable.sol:296-318, 161-178 | Статус: **Fixed**

Прочие пути изъятия фильтруют источник средств через `_requireNotProtected` (AccountIsVault / AccountIsRegulatoryEscrow / AccountIsLiquiditySafe). Однако функция `setRegulatoryEscrow` проверяет только нулевой адрес и address(this), а функция `seizeVaultEscrowToRegulatoryEscrow` не сопоставляет msg.sender с regulatoryEscrow.

Если администратор ошибочно установит regulatoryEscrow равным адресу vault (держателю VAULT_ROLE), внутренний вызов _transfer станет холостым переводом на самого себя, однако значение seizedEscrowBalance всё равно увеличится, что приведёт к рассинхронизации учёта; последующий вызов setRegulatoryEscrow будет заблокирован ошибкой EscrowDirty. Риск кражи отсутствует (cancelSeized защищён проверкой EscrowDrained); речь идёт о самопроизвольном нарушении учёта и доступности функционала.

Вероятность: низкая (ошибка конфигурации со стороны DEFAULT_ADMIN_ROLE; выявляется внеблокчейн-верификатором).

РЕКОМЕНДАЦИЯ

В функции setRegulatoryEscrow отклонять кандидата, для которого истинно hasRole(VAULT_ROLE) или hasRole(LIQUIDITY_SAFE_ROLE) (по аналогии с _requireNotProtected). Дополнительно в seizeVaultEscrowToRegulatoryEscrow рекомендуется добавить проверку msg.sender != regulatoryEscrow.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: setRegulatoryEscrow отклоняет кандидатов с VAULT_ROLE / LIQUIDITY_SAFE_ROLE; добавлена проверка msg.sender != regulatoryEscrow в seizeVaultEscrowToRegulatoryEscrow.

GLX-L-02

burn(address from, uint256) — избыточный параметр from

LOW | GLXTokenUpgradeable.sol:208-210 | Статус: **Fixed**

Единственный внешний вызов функции — `glx.burn(address(this), fineWeight)` из GoldVault.finalizeRedeem, то есть vault сжигает исключительно собственный эскроу. Сигнатура `burn(address from, ...)` позволяет держателю VAULT_ROLE сжечь токены любого адреса, что является избыточной поверхностью атаки (эксплуатация требует вредоносного обновления контракта vault).

Вероятность: низкая (роль VAULT_ROLE закреплена только за контрактом vault).

РЕКОМЕНДАЦИЯ

Изменить сигнатуру на burn(uint256 amount) со сжиганием с адреса msg.sender. Это не нарушает текущий единственный вызов. (Функция cancelSeized сжигает эскроу через внутренний _burn — изменения не требуются.)

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: сигнатура изменена на burn(uint256) со сжиганием с msg.sender; единственный вызов (finalizeRedeem) обновлён.

GLX-L-03

rescueERC20 запрещает вывод GLX с адреса самого токена

LOW | GLXTokenUpgradeable.sol:374-385 | Статус: **Acknowledged**

Функция rescueERC20 завершается ошибкой при token == address(this) (SelfRescue), поэтому токены GLX, ошибочно отправленные на адрес самого контракта токена, оказываются заблокированы безвозвратно. Это намеренная защитная проверка, предотвращающая нарушение учёта, однако она создаёт сценарий безвозвратной блокировки средств.

РЕКОМЕНДАЦИЯ

Сохранить защитную проверку (целостность учёта приоритетна) и задокументировать риск; в документации для интеграторов предупредить о недопустимости переводов GLX на адрес токена.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: защитная проверка SelfRescue сохранена намеренно (целостность учёта); задокументировано, интеграторов предупреждают о недопустимости переводов GLX на адрес токена.

GLX-I-01

Соответствие GoldVault.liquiditySafe и LIQUIDITY_SAFE_ROLE не обеспечивается на уровне контракта

INFO | GLXTokenUpgradeable.sol:467-473; GoldVaultUpgradeable.sol:575-599 | Статус: **Fixed**

Защита `_requireNotProtected` освобождает от заморозки и изъятия только держателей роли LIQUIDITY_SAFE_ROLE в контракте GLX. При этом GoldVault при депонировании эмитирует комиссию ликвидности на адрес из поля `liquiditySafe` и предварительно проверяет `glx.isFrozen(liquiditySafe)`, завершаясь ошибкой LiquiditySafeFrozen. Таким образом, защита привязана к РОЛИ, а эмиссия комиссии — к АДРЕСУ; равенство этих сущностей на уровне контракта не гарантируется.

Воздействие. Если адрес в поле liquiditySafe не является держателем LIQUIDITY_SAFE_ROLE (роль выдана другому адресу либо не выдана), этот адрес не защищён _requireNotProtected, и роль COMPLIANCE_ROLE может его заморозить. После этого при включённой комиссии (liquidityFeeBps > 0) каждый депозит завершается ошибкой LiquiditySafeFrozen — путь депонирования и эмиссии GLX оказывается заблокирован (нарушение доступности функционала, не кража). Состояние восстановимо приведением адреса и роли в соответствие либо обнулением комиссии. Это первопричина на уровне контракта для замечаний VLT-M-01 (грант роли пропущен в пакете назначения ролей) и VLT-L-01 (сеттер liquiditySafe не валидирует адрес).

Вероятность: низкая, операционная; риск латентен до включения комиссии.

РЕКОМЕНДАЦИЯ

Задокументировать инвариант «адрес liquiditySafe соответствует держателю LIQUIDITY_SAFE_ROLE» и добавить соответствующую проверку в verify_glx_role_topology; см. также рекомендации VLT-M-01 и VLT-L-01.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: инвариант усилен на уровне контракта (_requireLiquiditySafeReady) и в конфигурации верификатора.

GLX-I-02

seizeFrozenToEscrow генерирует событие FrozenFundsSeized при amount==0

INFO | GLXTokenUpgradeable.sol:248-268; docs/glx-token.md:128 | Статус: **Acknowledged**

Функция seizeFrozenToEscrow выполняет перевод только при amount > 0 (строка 259), однако событие FrozenFundsSeized генерируется безусловно (строка 267) — в том числе при нулевом балансе. Поведение не согласовано с «сестринской» функцией seizeFrozenAmountToEscrow, которая при нулевой сумме завершается ошибкой ZeroAmount, что создаёт избыточный шум в событиях.

Поведение предусмотрено спецификацией: docs/glx-token.md (раздел «Freeze and seizure», функция seizeFrozenToEscrow), пункт «Emits FrozenFundsSeized even when the balance is zero». Это утверждение в спецификации, а не комментарий в коде — в самой функции пояснение отсутствует. Замечание носит характер согласованности/наблюдаемости, дефектом не является.

РЕКОМЕНДАЦИЯ

Согласовать поведение двух функций либо отразить намеренность непосредственно в коде (поясняющим комментарием) дополнительно к спецификации.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: добавлен поясняющий комментарий — генерация события при нулевой сумме намеренна (аудит-якорь под orderHash).

GLX-I-03

Переопределение _approve блокирует approve/permit при паузе

INFO | GLXTokenUpgradeable.sol:428-448 | Статус: **Acknowledged**

Блокировка ненулевых вызовов approve/permit при паузе отклоняется от ряда устоявшихся практик DeFi. Частично компенсируется механизмом AV4-L-02 (вызов approve(_, 0) разрешён всегда). С высокой вероятностью предусмотрено архитектурой.

РЕКОМЕНДАЦИЯ

Подтвердить намеренность поведения; при необходимости ослабить блокировку approve при паузе.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: добавлен поясняющий комментарий — блокировка ненулевого approve при паузе намеренна (circuit-breaker; approve(_, 0) остаётся доступным).

5.2 GoldBarNFTUpgradeable.sol

GBAR-L-01

Отзыв одобрения склада блокирует уже выпущенный слиток до его депонирования

LOW | GoldBarNFTUpgradeable.sol:187 / GoldVaultUpgradeable.sol:555-558 | Статус: **Acknowledged**

Функция mintBar проверяет approvedWarehouses при выпуске, а onERC721Received выполняет проверку одобрения повторно при депонировании. Если роль DEFAULT_ADMIN_ROLE отзовёт одобрение склада в промежутке между выпуском и депонированием, держатель не сможет внести слиток и получить GLX; NFT останется в состоянии Minted (передаваемом), но непригодным для обеспечения, что приведёт к расхождению с физическим слитком вне блокчейна. Состояние восстановимо повторным одобрением склада.

Вероятность: низкая (требует отзыва одобрения склада при наличии невнесённых слитков).

РЕКОМЕНДАЦИЯ

Сохранять право на депонирование для ранее выпущенных слитков (фиксация допустимости на момент выпуска) либо на операционном уровне гарантировать, что одобрение склада не отзывается при наличии активных слитков в состоянии Minted. При этом проверку одобрения склада на этапе депонирования необходимо сохранить — это комплаенс-контроль; устранять проблему за счёт её ослабления (приёма слитков любого склада) недопустимо.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: операционный инвариант — одобрение склада не отзывается, пока есть слитки в состоянии Minted (предварительная проверка состояния перед revoke).

GBAR-L-02

markDeliveredAndCancel для несуществующего токена возвращает типовую ошибку OpenZeppelin

LOW | GoldBarNFTUpgradeable.sol:355 | Статус: **Fixed**

Функция markDeliveredAndCancel читает ownerOf(tokenId) до любого вызова _transition. Для несуществующего токена ownerOf из OpenZeppelin завершается строковой ошибкой раньше, чем сработает пользовательская ошибка NonexistentToken. Влияние на безопасность отсутствует.

РЕКОМЕНДАЦИЯ

Добавить в начало функции проверку if (!_exists(tokenId)) revert NonexistentToken(); либо принять типовую ошибку OpenZeppelin и задокументировать это.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: добавлена проверка _exists(tokenId) с пользовательской ошибкой NonexistentToken.

GBAR-L-03

Корректность releaseTo опирается на инвариант «vault не может быть заморожен»

LOW | GoldBarNFTUpgradeable.sol:406 ↔ GoldVaultUpgradeable.sol:488 | Статус: **Acknowledged**

Хук _beforeTokenTransfer применяет проверки isFrozen(from)/isFrozen(to), в том числе к ограниченному переводу vault → получатель при освобождении слитка. Если бы vault оказался заморожен в контракте GLXToken, releaseTo завершался бы ошибкой FromFrozen(vault), блокируя finalizeRedeem (без потери средств, но с потерей работоспособности процесса погашения). В настоящее время инвариант соблюдается: _requireNotProtected не допускает заморозку держателя VAULT_ROLE.

Проверка заморозки источника (from) на пути освобождения не приносит пользы (vault является доверенным), но создаёт поверхность для отказа в обслуживании при нарушении инварианта.

РЕКОМЕНДАЦИЯ

Задокументировать зависимость (не отзывать VAULT_ROLE у vault, пока он способен принимать NFT). Дополнительно: пропускать проверки заморозки источника и оператора, когда перевод соответствует открытому окну освобождения.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: операционный инвариант — VAULT_ROLE не отзывается у vault при наличии незавершённых запросов на погашение.

GBAR-L-04

_beforeTokenTransfer не проверяет batchSize==1

LOW | GoldBarNFTUpgradeable.sol:386-427 | Статус: **Fixed**

Проверки состояния и блокировки (строки 419-426) выполняются только для firstTokenId. В OpenZeppelin v4 ERC-721 данный хук вызывается с batchSize==1 (пакетная обработка имеется лишь в неиспользуемом ERC721Consecutive), поэтому в текущей реализации поведение безопасно. Будущее обновление с пакетным выпуском или переводом обойдёт проверки для элементов с индексом больше нулевого.

РЕКОМЕНДАЦИЯ

Добавить проверку batchSize == 1 (откат при значении больше единицы) либо инвариант в validate-upgrade, фиксирующий отсутствие пакетных переводов.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: добавлен откат BatchTransferUnsupported при batchSize != 1.

GBAR-I-01

BarMetadata хранит параметры, принудительно равные константам

INFO | GoldBarNFTUpgradeable.sol:173-210 | Статус: **Acknowledged**

Функция mintBar принимает grossWeight и fineness, проверяет их на равенство констант STANDARD_GROSS_WEIGHT и STANDARD_FINENESS и затем сохраняет; значение fineWeight вычисляется из них. Поскольку входные значения принудительно равны константам, сохранённые поля всегда равны константам, а fineWeight всегда составляет 999.9·10^18. Прямое использование констант сократило бы расход газа и повысило читаемость.

Оговорка: хранение значений индивидуально для каждого слитка обеспечивает совместимость с будущей версией, поддерживающей разнородные слитки (см. VLT-I-02); поэтому замечание имеет низкий приоритет — оптимизация уместна лишь в случае отказа от такой совместимости в текущей версии.

РЕКОМЕНДАЦИЯ

Использовать константы напрямую (если совместимость с разнородными слитками в данной версии не требуется) либо сохранить текущую схему и задокументировать намерение «структура рассчитана на будущие разнородные слитки».

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: поля BarMetadata сохранены индивидуально для совместимости с будущей версией, поддерживающей разнородные слитки.

5.3 GoldVaultUpgradeable.sol

VLT-M-01

Пакет назначения ролей не выдаёт LIQUIDITY_SAFE_ROLE — риск блокировки депозитов

MEDIUM | role_wiring_glx_core_upgradeable.glx.example.json; GoldVault.sol:575-599 | Статус: **Fixed**

Защита адреса liquiditySafe от заморозки реализована на уровне ролей: _requireNotProtected в контракте GLXToken освобождает от заморозки и изъятия только держателей LIQUIDITY_SAFE_ROLE. Конфигурация deploy_gold_vault_proxy.glx.example.json (строки 25-26) прямо требует выдать эту роль адресу liquiditySafe в составе пакета назначения ролей. Однако фактический пакет назначения ролей выдаёт только VAULT_ROLE и семь операционных ролей — грант LIQUIDITY_SAFE_ROLE отсутствует.

При последующем включении комиссии (setLiquidityFeeBps > 0) роль COMPLIANCE_ROLE сможет выполнить freeze(liquiditySafe), после чего каждый депозит с комиссией будет завершаться ошибкой LiquiditySafeFrozen — весь путь депонирования и эмиссии GLX окажется заблокирован. Встроенная защитная проверка лишь преобразует сбой в корректный откат, но не предотвращает заморозку.

Воздействие: блокировка эмиссии (доступность функционала), не кража. **Вероятность:** от низкой до средней (по умолчанию комиссия равна нулю, поэтому риск латентен до её включения).

РЕКОМЕНДАЦИЯ

1) Добавить в пакет назначения ролей грант LIQUIDITY_SAFE_ROLE для адреса liquidity_safe. 2) Добавить проверку hasRole(LIQUIDITY_SAFE_ROLE, liquiditySafe) в verify_glx_role_topology и в контрольный список. 3) Превентивное усиление: завершать ошибкой setLiquidityFeeBps/setLiquiditySafe (и initialize при комиссии больше нуля), если роль не выдана.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: грант LIQUIDITY_SAFE_ROLE добавлен в пакет назначения ролей; добавлена проверка _requireLiquiditySafeReady в initialize / setLiquiditySafe / setLiquidityFeeBps; ожидание роли добавлено в конфигурацию верификатора.

VLT-L-01

setLiquiditySafe не отклоняет заморожённый адрес или адрес без требуемой роли

LOW | GoldVaultUpgradeable.sol:239-251 | Статус: **Fixed**

Функция setLiquiditySafe проверяет лишь нулевой адрес, совпадение с собой и отсутствие изменения; она не проверяет glx.isFrozen и наличие LIQUIDITY_SAFE_ROLE — фактические предусловия успешной эмиссии комиссии. Назначение «некорректного» адреса при ненулевой комиссии проявится только при следующем депозите, завершающемся ошибкой. Связано с VLT-M-01.

РЕКОМЕНДАЦИЯ

Завершать ошибкой при glx.isFrozen(newLiquiditySafe) и (опционально) если новый адрес не является держателем LIQUIDITY_SAFE_ROLE — перенос инварианта депозита на этап конфигурации.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: setLiquiditySafe вызывает _requireLiquiditySafeReady (проверка заморозки и наличия роли).

VLT-L-02

Поле redemptionLocked в DepositRecord нигде не читается (неиспользуемое хранилище)

LOW | GoldVaultUpgradeable.sol:49 | Статус: **Fixed**

Поле deposit.redemptionLocked только записывается и нигде не читается. Блокировка отслеживается состоянием NFT (RedemptionLocked) и через значения _lockedFineWeight / request.state.

РЕКОМЕНДАЦИЯ

Контракты развёрнуты только в тестовой сети, поэтому удалить поле из структуры DepositRecord в исходном коде и повторно развернуть тестовую сеть до выхода в mainnet. После выхода в mainnet удаление поля (изменение раскладки структуры внутри mapping) потребовало бы обновления с валидацией раскладки (validate-upgrade).

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: неиспользуемое поле redemptionLocked удалено; эталоны раскладки перегенерированы.

VLT-L-03

_eligibleIndexOf хранит uint256, тогда как достаточно булевой семантики

LOW | GoldVaultUpgradeable.sol:80, 653-671 | Статус: **Fixed**

Значение _eligibleIndexOf нигде не читается как индекс (очередь функционирует через _eligibleAt и указатели head/tail); используется только проверка на неравенство нулю (признак членства). Текущая реализация корректна.

РЕКОМЕНДАЦИЯ

Контракты развёрнуты только в тестовой сети (подтверждено клиентом), поэтому исправление следует внести сейчас, до выхода в mainnet: заменить тип на mapping(uint256 => bool) в исходном коде и повторно развернуть тестовую сеть (тестовое развёртывание не несёт реальных средств, и сохранять его раскладку не требуется). После выхода в mainnet такое изменение раскладки потребовало бы обновления с валидацией раскладки

(validate-upgrade). Если планируется удаление произвольного элемента очереди за время O(1) (swap-and-pop), сохранить тип uint256 (index+1) и задокументировать назначение поля.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: тип изменён на mapping(uint256 => bool) (_isEligible).

VLT-I-01

Комиссия ликвидности: для погашения требуется полный fineWeight (предусмотрено архитектурой)

INFO | GoldVaultUpgradeable.sol:331-372, 538-604 | Статус: **Acknowledged**

При liquidityFeeBps > 0 вкладчик получает fineWeight − feeAmount GLX, однако функции requestRedeem/redeemBatch требуют полного значения fineWeight. Таким образом, вкладчик не может погасить тот же слиток без приобретения недостающего feeAmount на рынке (зависимость от ликвидности liquiditySafe). При этом инвариант totalSupply == backingFineWeight сохраняется. Дефекты отсутствуют.

РЕКОМЕНДАЦИЯ

Сделать функцию previewDeposit заметной в клиентском приложении и добавить в docs/glx-core.md примечание о том, что погашение требует полного значения fineWeight независимо от удержанной комиссии.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: модель комиссии задокументирована — для погашения требуется полный fineWeight.

VLT-I-02

Допущение однородности слитков: FIFO-выбор корректен только при едином профиле

INFO | GoldVaultUpgradeable.sol:331-372, 653-671; GoldBarNFTUpgradeable.sol:173-210 | Статус: **Acknowledged**

Текущая версия жёстко фиксирует единый профиль слитка (grossWeight=1000·10^18, fineness=9999, соответственно fineWeight=999.9·10^18), а Vault выбирает слиток для погашения строго по принципу FIFO, без возможности выбора со стороны пользователя. Это корректно и безопасно, пока все слитки однородны.

Если будущая версия введёт разнородные слитки (различные fineWeight и иные параметры) — что прямо предусмотрено docs/glx-core.md (раздел Economic model: поддержка разных весов «should be introduced through a new version of the specification and contracts»), — модель FIFO станет неадекватной: пользователь не сможет выбрать нужный слиток, а очередь способна приводить к застреванию запросов или неоптимальному выбору (поведение, подобное отказу в обслуживании для отдельных запросов).

Это НЕ дефект текущего кода и формально находится вне скоупа (аудит покрывает текущую реализацию, а не гипотетические будущие обновления). Зафиксировано как явное проектное допущение для следующей версии.

РЕКОМЕНДАЦИЯ

При переходе к разнородным слиткам пересмотреть модель выбора в Vault (например, выбор по конкретному запросу или набору вместо чистого FIFO) и провести отдельный аудит этой

версии.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: зафиксировано как проектное допущение для будущей версии (разнородные слитки).

5.4 Инструментарий обновлений и пакеты развёртывания

TL-M-01

Генератор пакетов Timelock не проверяет равенство администратора прокси целевому TimelockController

MEDIUM | generate_timelock_upgrade_bundle_glx.ts:280; safe-bundle-common.ts:408-434 | Статус: **Fixed**

Ключевой принцип модели (M1): администратором прокси является TimelockController, а Upgrader Safe администратором не является. Генератор проверяет совместимость хранилища и ненулевой слот реализации, но не читает слот администратора EIP-1967 и не проверяет его равенство timelockAddress. Пакет может быть сгенерирован для прокси с неверным TimelockController или с администратором-EOA: операция schedule пройдёт успешно, оператор выждет minDelay, а вызов execute завершится откатом на этапе исполнения (прокси отвергнет вызов не от администратора) — что приводит к потере окна задержки; в условиях аварийной миграции это реальный ущерб.

Вероятность: от низкой до средней (ошибочное копирование неверного алиаса TimelockController среди четырёх схожих шаблонов). Тест GLXCoreUpgrade.test.ts проверяет равенство администратора и TimelockController, но генератор — нет.

РЕКОМЕНДАЦИЯ

Читать слот администратора EIP-1967 и проверять его равенство cfg.timelockAddress; прерывать генерацию при несовпадении (по аналогии с проверкой слота реализации).

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: генератор читает EIP-1967 admin-слот и прерывается при несоответствии TimelockController.

TL-M-02

verify_glx_role_topology не проверяет инвариант «администратор прокси = TimelockController»

MEDIUM | verify_glx_role_topology_glx.ts:373-473 | Статус: **Fixed**

Верификатор, выполняемый после развёртывания (контрольный рубеж перед выходом в mainnet), проверяет роли, заморозку эскроу и наличие кода, но не читает слот администратора трёх прокси. Развёртывание, в котором администратором прокси по ошибке остался Upgrader Safe, адрес-EOA или неверный TimelockController, пройдёт верификацию, скрыто нарушая модель управления обновлениями. В конфигурации отсутствует соответствующее поле.

РЕКОМЕНДАЦИЯ

Добавить блок proxyAdmins:[{label, proxy, expectedAdmin}] и проверять равенство getStorageAt(proxy, ADMIN_SLOT) и TimelockController для каждого прокси.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: верификатор проверяет admin-слот всех трёх прокси (блок proxyAdmins).

TL-M-03

validate-upgrade.ts работает по принципу fail-open: пропускает цели/эталоны, сообщая об успехе

MEDIUM | validate-upgrade.ts:107-118, 142-159; .github/workflows/glx-core.yml | Статус: **Fixed**

В конвейере CI переменная REQUIRE_UPGRADE_BASELINES=1 намеренно не установлена. Отсутствующий артефакт приводит к предупреждению и пропуску цели; отсутствующий эталон — к сообщению и пропуску. При этом скрипт всё равно выводит «All upgrade-safety checks passed» и завершается с кодом 0. Переименованный или удалённый артефакт даёт успешный результат проверки совместимости, фактически не проверив ничего. Строгий режим является опциональным и не используется.

Вероятность: средняя (любой рефакторинг, меняющий имя артефакта; режим CI по умолчанию нестрогий).

РЕКОМЕНДАЦИЯ

Установить REQUIRE_UPGRADE_BASELINES=1 в задаче CI на полном репозитории; как минимум — завершать ошибкой (а не предупреждением) при отсутствии артефакта или эталона любой из трёх целей в скоупе.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: fail-closed для целей GLX Core (strict) и REQUIRE_UPGRADE_BASELINES=1 в CI.

CFG-M-01

Конфигурация верификатора не содержит LIQUIDITY_SAFE_ROLE — пропуск гранта не выявляется

MEDIUM | verify_glx_role_topology.glx.example.json:44-105 | Статус: **Fixed**

Дополняет VLT-M-01: верификатор не содержит ожидания по LIQUIDITY_SAFE_ROLE, поэтому даже полный запуск не выявит пропущенный грант. Это единственная защищённая роль, которую верификатор не проверяет (VAULT_ROLE и regulatoryEscrow проверяются).

РЕКОМЕНДАЦИЯ

Добавить ожидание LIQUIDITY_SAFE_ROLE (держатель — Liquidity Safe) в блок GLXToken верификатора, а соответствующий грант — в пакет назначения ролей.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: грант LIQUIDITY_SAFE_ROLE в пакете назначения ролей и ожидание роли в конфигурации верификатора.

TL-L-01

Эталоны хранилища являются снимком текущих контрактов

LOW | src/ts/upgrade-references/*.reference.json; validate-upgrade.ts:123-141 | Статус: **Acknowledged**

Зафиксированные в репозитории эталоны представляют собой артефакты ТЕКУЩИХ реализаций, а не предыдущей версии. Функция validateUpgrade сравнивает контракт сам с собой и не может выявить несовместимость, пока кто-либо не изменит контракт без обновления эталона. Защита проявляется только при будущем изменении. Дополнительно: артефакты не содержат поля storageLayout — OpenZeppelin восстанавливает раскладку из данных компиляции по байт-коду.

РЕКОМЕНДАЦИЯ

Задокументировать, что эталон должен соответствовать РАНЕЕ РАЗВЁРНУТОЙ реализации. Добавить в CI проверку того, что байт-код эталона отличается от рабочего дерева при изменении исходника.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: эталоны перегенерированы под текущие контракты; ротация эталонов описана в runbook.

TL-L-02

Расхождение обработки соли CREATE2: предсказатель дополняет нулями, генератор отклоняет

LOW | safe-bundle-common.ts:287-301; predict_glx_create2_addresses.ts:91-95 | Статус: **Fixed**

Функция computeCreate2Address дополняет соль нулями через hexZeroPad(salt, 32), тогда как генератор кодирует исходную соль в performCreate2(bytes32). Некорректную соль предсказатель молча дополнит нулями, а генератор отклонит на этапе кодирования — в результате предсказатель может выдать адрес, который пакет никогда не сформирует. Текущие конфигурации используют корректные 32-байтовые соли, фактического расхождения нет.

РЕКОМЕНДАЦИЯ

Валидировать соль на точное соответствие 32 байтам (requireBytes32) в обоих генераторах и в предсказателе.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Fixed: единая строгая проверка 32-байтовой соли (requireBytes32) в генераторе и предсказателе.

CFG-L-01

Шаблон обновления с миграцией: эталон именуется по контракту V2, но должен содержать раскладку V1

LOW | upgrade_glx_token_with_migration.glx.example.json; safe-bundle-common.ts:341-392 | Статус: **Acknowledged**

Параметр deploy.contract задан как GLXTokenUpgradeableV2; проверка ищет эталон по этому имени, которое отсутствует, поэтому генерация всегда завершается ошибкой «Missing baseline». При создании эталона файл именуется по новому контракту (V2), однако для осмысленного validateUpgrade он должен содержать раскладку старого (V1) — это создаёт риск ошибочно поместить раскладку V2 и получить сравнение контракта с самим собой.

РЕКОМЕНДАЦИЯ

Связывать эталон с прокси или логическим именем; либо документировать, что эталон должен содержать раскладку V1, и выводить в журнал, какую раскладку представляет эталон.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: шаблон обновления с миграцией — намеренный placeholder; без эталона генерация прерывается (fail-closed).

TL-I-01

Режим createMode:"create" не регистрирует алиасы (предусмотрено: используется только create2)

INFO | generate_safe_bundle_glx.ts:196-238, 282-322 | Статус: **Acknowledged**

Регистрация алиасов выполняется только в ветках create2. Пакет с createMode:create и зависимостями {ref} завершится ошибкой «Unknown alias reference». Поскольку для GLX повсеместно требуется create2, в штатном сценарии ситуация недостижима.

РЕКОМЕНДАЦИЯ

Отклонять createMode:create для пакетов с зависимостями {ref} либо задокументировать ограничение.

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: режим create2-only предусмотрен архитектурой.

TL-I-02

Верификатор сопоставляет роли по именам необновляемых артефактов

INFO | verify_glx_role_topology.glx.example.json:45,107,165 | Статус: **Acknowledged**

Конфигурация указывает contract: GLXToken/GoldBarNFT/GoldVault, тогда как развёрнуты варианты *Upgradeable. Это работает, пока сигнатуры hasRole и геттеров совпадают между артефактами; расхождение ABI привело бы к скрытому некорректному декодированию.

РЕКОМЕНДАЦИЯ

*Указывать в конфигурации фактически развёрнутые артефакты *Upgradeable.*

РЕШЕНИЕ КОМАНДЫ РАЗРАБОТКИ

Acknowledged: верификатор поставляется как reference-source; боевая конфигурация использует имена *Upgradeable.

---

6. Заключение

GLX Core представляет собой зрелую систему с подробной спецификацией; критических дефектов логики не обнаружено, системные инварианты подтверждены фаззинг-тестированием и регрессионными наборами. По итогам раунда исправлений все замечания уровня MEDIUM устранены, а замечания уровня LOW/INFO исправлены либо осознанно приняты; открытых пунктов не осталось. Корректность исправлений подтверждена локальным прогоном тестов исправленной версии без отказов и без регрессий после изменения раскладки хранилища.

Итог раунда исправлений

- **VLT-M-01 / CFG-M-01** — устранено: грант LIQUIDITY_SAFE_ROLE добавлен в пакет назначения ролей и в проверки верификатора; защита продублирована на уровне контракта.
- **TL-M-01 / TL-M-02** — устранено: проверка равенства администратора прокси и TimelockController добавлена в генератор пакетов и в верификатор.
- **TL-M-03** — устранено: проверка безопасности обновлений переведена в режим fail-closed.
- **GLX-M-01** — принято: централизация роли COMPLIANCE зафиксирована как осознанная модель регулируемого актива с операционными обязательствами.

Выше перечислены ключевые исправления уровня MEDIUM. Полный статус по каждой находке приведён в разделе 4 (сводный реестр) и разделе 5 (детальные находки): 15 — Fixed, 13 — Acknowledged, 0 — Open.

Рекомендации на следующий цикл

- Поддерживать операционные инварианты из принятых замечаний в регламенте эксплуатации (отзыв одобрений складов, ротация ролей VAULT_ROLE и Liquidity Safe).
- Анализ полноты тестового покрытия (путь OperatorFrozen в GoldBarNFT; граничные сценарии redeemBatch; негативные сценарии инструментария).
- Повторный аудит при переходе к разнородным слиткам (см. VLT-I-02) — модель выбора в Vault потребует пересмотра.