ПЛАН РЕАГИРОВАНИЯ НА ИНЦИДЕНТЫ БЕЗОПАСНОСТИ ДЛЯ ВЕБ-СЕРВЕРА

Профилактика инцидентов

### Базовые меры безопасности.

Использование TLS 1.3 с современными шифрами (например, AES-256-GCM). Автоматическое обновление сертификатов через Let’s Encrypt. Настройка WAF (ModSecurity, Cloudflare) для блокировки SQLi, XSS и других атак. Ограничение входящего/исходящего тракта через iptables или nftables. Закрыть неиспользуемые порты и удалить дефолтные аккаунты.

### Регулярное обновление.

Автоматические обновления безопасности для ОС.Обновление веб-сервера (Nginx/Apache), СУБД и языков (PHP/Python). Сканирование на уязвимости в библиотеках (Snyk, OWASP Dependency-Check). Использование только доверенных репозиториев (например, официальные npm/pip).

Контроль доступа.

MFA для административных панелей. Блокировка IP после 5 неудачных попыток входа (Fail2Ban). Принцип минимальных привилегий для пользователей и сервисов. Отдельные учётные записи для CI/CD и деплоя. Отключение входа по паролю (только ключи Ed25519). Смена дефолтного порта SSH (не 22).

Резервное копирование.

Ежедневные бэкапы БД и файлов. Хранение копий в изолированном хранилище (S3 с Versioning). Регулярная проверка восстановления из бэкапа. Шифрование резервных копий (GPG, AES).

Мониторинг и логирование.

Централизованный сбор логов (ELK, Grafana Loki). Алерты на подозрительные события. Detectify для поиска уязвимостей. Мониторинг аномалий в трафике.

Конфигурация безопасности.

Запрет выполнения скриптов в upload-директориях. Отключение лишних модулей веб-сервера. Использование Docker с read-only файловой системой. Изоляция сервисов через namespaces и cgroups.

Обучение и тестирование.

Аудиты по OWASP Top 10 и безопасному кодированию и статический анализ кода (SonarQube, Semgrep) для разработчиков. Тренировки по реагированию на инциденты (Red Team vs Blue Team) и регулярные пентесты (Metasploit, Burp Suite) для администраторов.

Реагирование на инциденты.

Чек-лист для блокировки атак. Изоляция зараженных нод. Контакты ответственных лиц. Шаблоны уведомлений для пользователей (в случае утечки данных).

Защита от утечек.

Хранение API-ключей и паролей в Vault или AWS Secrets Manager. Сканирование Git-репозиториев на утечки (TruffleHog, GitGuardian). Маскирование чувствительных данных в логах.

Дополнительные инструменты.

Nikto для поиска уязвимостей конфигурации. OpenVAS для сетевого аудита. ClamAV для проверки загружаемых файлов. RKHunter для поиска руткитов.

Обнаружение

Сломанный контроль доступа (Broken Access Control): автоматизированное тестирование, ручная проверка прав через изменение параметров запросов и анализ логов на подозрительные запросы.

Криптографические сбои (Cryptographic Failures): сканирование TLS/SSL, Проверка наличия шифрования для конфиденциальных данных.

Инъекции (Injection): сканеры уязвимостей, мониторинг логов на аномальные запросы.

Небезопасный дизайн (Insecure Design): threat modeling на этапе проектирования, тестирование сценариев злоупотребления бизнес-логикой.

Ошибки конфигурации (Security Misconfiguration): инструменты аудита и сканирование уязвимостей.

Уязвимые и устаревшие компоненты: сканирование зависимостей.

Ошибки аутентификации и идентификации: анализ частоты неудачных входов, проверка наличия блокировок после множества попыток.

Сбои в обеспечении целостности программного обеспечения и данных: проверка цифровых подписей, мониторинг изменений в CI/CD.

Сбои в работе журналов безопасности и мониторинга: интеграция с SIEM-системами, регулярный аудит логов на аномалии.

Подделка запросов на стороне сервера (SSRF): валидация входящих URL, мониторинг исходящих запросов сервера.

Способы защиты

Сломанный контроль доступа (Broken Access Control): реализация принципа минимальных привилегий (RBAC/ABAC), проверка прав доступа на уровне API и UI, использование UUID вместо предсказуемых user\_id, регулярный аудит ACL (Access Control Lists).

Криптографические сбои (Cryptographic Failures): использование современных алгоритмов (AES-256, SHA-3, Argon2), принудительное шифрование данных (HTTPS с TLS 1.3), хранение засекреченных данных (пароли, ключи) в защищённых хранилищах (HashiCorp Vault, AWS KMS), регулярная ротация сертификатов (например, через Let’s Encrypt).

Инъекции (Injection): санитизация и валидация всех входных данных (регулярные выражения, whitelist), использование параметризованных запросов (ORM, Prepared Statements), экранирование специальных символов (например, HTML-сущностей для XSS), внедрение CSP (Content Security Policy) для блокировки вредоносных скриптов.

Небезопасный дизайн (Insecure Design): внедрение Threat Modeling на этапе проектирования, тестирование сценариев злоупотребления (например, лимиты на транзакции), использование шаблонов безопасной архитектуры (Zero Trust, Defense in Depth).

Ошибки конфигурации (Security Misconfiguration): автоматизация настройки инфраструктуры (Infrastructure as Code: Terraform, Ansible), регулярный аудит конфигураций (например, CIS Benchmarks), отключение ненужных сервисов и портов, использование "закаленных" образов (например, Docker с минимальными правами).

Уязвимые и устаревшие компоненты: автоматическое обновление зависимостей (Dependabot, RenovateBot), мониторинг CVE через инструменты (Snyk, OWASP Dependency-Check), использование подписанных пакетов (например, GPG-подписи в npm).

Ошибки аутентификации и идентификации: внедрение MFA (Google Authenticator, YubiKey), блокировка аккаунтов после N неудачных попыток, использование OAuth 2.0/OpenID Connect для делегированной аутентификации, хранение паролей в хешированном виде с "солью" (bcrypt, scrypt).

Сбои в обеспечении целостности программного обеспечения и данных: цифровая подпись пакетов и обновлений, контроль целостности файлов (например, Tripwire), использование приватных репозиториев с доступом по SSH-ключам (например, Nexus, Artifactory).

Сбои в работе журналов безопасности и мониторинга: централизованный сбор логов (ELK Stack, Graylog), настройка алертов на подозрительные события, регулярное тестирование мониторинга через "красные команды" (Red Team exercises).

Подделка запросов на стороне сервера (SSRF): блокировка запросов к внутренним IP-адресам и метаданным (AWS IMDSv2), валидация URL через whitelist домены, использование прокси для исходящих запросов с фильтрацией.