Муниципальное автономное общеобразовательное учреждение
"Гимназия № 10" Пушкинского муниципального района Московской области

Проект: «Бот для автоматизации отчетности по информационной безопасности в Telegram»

Ученик **Огурцова Мария, 10 класса**

Москва - 2025

Оглавление

ИСПОЛЬЗУЕМЫЕ ТЕРМИНЫ И СОКРАЩЕНИЯ

Бот (Telegram-бот) — программный агент, работающий через Telegram Bot API и выполняющий автоматические операции (приём команд, отправка сообщений, планирование задач) без участия пользователя в интерфейсе мессенджера.

1. Лог (лог-файл, журнал) — файл или поток записей, содержащих системные события, сообщения приложений и другие данные о работе информационной системы; используется для анализа и расследования инцидентов.

2. Событие (Event) — единичная запись в логе, описывающая конкретное действие или состояние системы (например, попытка входа, ошибка сервера, сетевое соединение).

3. Нормализация (Normalization) — процесс приведения разнородных записей событий к единому структуированному формату (например, {timestamp, source, level, message, metadata}) для последующей агрегации и анализа.

4. Агрегация (Aggregation) — вычисление статистик и показателей (KPI) по набору событий за заданный период: подсчёт количества событий, распределение по уровням, выделение топ-сообщений и пр.

5. KPI (Key Performance Indicator) — ключевой показатель эффективности; в контексте проекта: метрики ИБ, например, количество инцидентов за период, среднее время реакции, число критических уязвимостей.

6. Отчёт (Report) — структурированный документ (в проекте — в формате Markdown), содержащий резюме состояния безопасности за период, ключевые метрики, перечень инцидентов и рекомендации.

7. Шаблон отчёта (Report Template) — заранее заданная структура и формат представления данных в генерируемом отчёте (заголовки, секции, порядок вывода метрик).

8. Расписание (Schedule) — конфигурация планировщика задач, задающая периодичность и время автоматической генерации и отправки отчётов (например, ежедневное распространение в 09:00).

9. JobQueue — планировщик задач, встроенный в библиотеку python-telegram-bot (или аналогичная компонентная реализация), используемый для периодического запуска функций генерации отчётов.

10. sqlite3 — встроенная реляционная встраиваемая база данных, используемая для хранения конфигурации бота, расписаний и журнала отправленных отчётов.

11. Collector — модуль проекта, отвечающий за сбор данных из источников (файлы логов, локальные сервисы, внешние API).

12. Normalizer — модуль проекта, выполняющий преобразование собранных данных в унифицированную структуру.

13. Aggregator — модуль проекта, выполняющий вычисление KPI и подготовку агрегированных данных для отчёта.

14. ReportGenerator (report_gen) — модуль проекта, формирующий итоговый отчёт в формате Markdown на основе агрегированных данных и шаблона.

15. Источник данных (Data Source) — система или файл, из которого считываются события: системные логи, логи приложений, логи сетевых устройств, ответы внешних API сканирования уязвимостей и т.п.

16. Инцидент (Incident) — событие или серия событий, свидетельствующих о нарушении политики безопасности или о фактическом или потенциальном компрометировании системы.

17. Анонимизация (Anonymization) — процесс удаления или замены персональных и идентифицирующих данных в логах и отчётах для соблюдения требований конфиденциальности. В контексте проекта применяется к следующим категориям данных: IP-адреса, имена пользователей, email-адреса, номера телефонов и другие идентификаторы. Методы: замена на обобщенные значения (например, [REDACTED]), маскирование части данных (192.168.xxx.xxx) или полное удаление поля.

18. TLS (Transport Layer Security) — криптографический протокол, обеспечивающий защищённую передачу данных при взаимодействии с внешними API и сервисами.

19. API (Application Programming Interface) — интерфейс взаимодействия с внешними сервисами (например, сервисами сканирования уязвимостей), используемый для получения метрик и событий.

20. Compliance / Нормативно-правовые требования — совокупность законодательных и корпоративных требований к обработке данных и ведению журналов (например, требования по хранению и защите персональных данных).

21. Парсинг (Parsing) — процесс синтаксического разбора строк логов с целью извлечения полей и значений (timestamp, уровень, сообщение и пр.).

22. Тестовый чат — приватный или групповой Telegram-чат, используемый для проверки работы бота и отправки тестовых отчётов в процессе разработки.

23. Отчётность по информационной безопасности — регулярное представление результатов мониторинга ИБ заинтересованным лицам (администраторам, менеджерам), включающее метрики, инциденты и рекомендации.

24. UI (User Interface) команд бота --- набор текстовых команд Telegram (например, /send_now, /set_report, /status), предоставляющих пользователю управление функционалом бота. Доступ к административным командам ограничен списком разрешенных user_id, хранящимся в базе данных.

25. CSV (Comma-Separated Values) — текстовый формат для обмена табличными данными; возможный формат экспорта отчётов.

26. SLA (Service Level Agreement) — соглашение об уровне сервиса; в контексте проекта подразумевает целевые показатели доступности и своевременности отправки отчётов (например, 95% успешных отправок).

27. Сниффинг (Sniffing) — перехват сетевого трафика; упоминается в контексте угроз, потенциально выявляемых через анализ логов.

28. IP-адрес — числовой адрес устройства в сети (IPv4/IPv6), используемый для идентификации источников событий и корреляции инцидентов.

29. UA (User-Agent) — строка, идентифицирующая клиентское приложение (браузер, OS), используемая для дополнительной фильтрации и анализа логов.

Введение

Информационная безопасность сегодня — это не абстрактная тема, а повседневная необходимость: число кибератак, утечек и эксплойтов растёт, а организации любого масштаба сталкиваются с риском потерь данных и нарушений доступности сервисов. Малые предприятия и учебные сети особенно уязвимы: у них нет ресурсов на дорогостоящие SIEM-решения, а ручная отчётность по событиям и уязвимостям занимает много времени и даёт низкую оперативность реагирования.

В последние годы внедрение ботов в мессенджерах, таких как Telegram, стало популярной практикой среди разработчиков как способ автоматизации рутинных задач. Такие инструменты могут значительно облегчить процессы сбора и доставки информации. Мониторинг состояния серверов и инфраструктуры становится критически важным для своевременного выявления угроз и инцидентов. В условиях быстроменяющейся киберугрозы, эффективный мониторинг позволяет не только минимизировать риски, но и сократить время реагирования на потенциальные инциденты.

Проблема: в условиях ограниченных ресурсов сбор и анализ логов, агрегация метрик и регулярное информирование ответственных сотрудников часто выполняются вручную или "на коленке". Это ведёт к задержкам в обнаружении инцидентов, непоследовательной отчётности и риску пропуска критических событий. Актуальность: автоматизация процесса формирования и доставки отчётов по информационной безопасности снижает время реакции на инциденты, повышает прозрачность состояния системы и делает мониторинг доступным для малого бизнеса и образовательных учреждений. Использование мессенджера Telegram как канала доставки обеспечивает удобство и быстроту получения информации ответственными лицами.

Цель проекта: разработать и внедрить Telegram-бота, который автоматизирует сбор, нормализацию и агрегацию данных о состоянии безопасности и формирует регулярные человеко-читаемые отчёты с возможностью гибкого управления расписанием и содержанием.

Задачи

1. Спроектировать архитектуру решения и структуру хранилища (sqlite3) для метаданных и истории отчётов.

2. Реализовать модуль сбора данных (collector) из локальных логов и (при необходимости) внешних API.

3. Реализовать модуль нормализации (normalizer) для приведения событий к единому формату с применением политик анонимизации конфиденциальных данных.

4. Разработать агрегатор (aggregator) для вычисления KPI и выявления ключевых инцидентов.

5. Создать генератор отчётов (report_gen) в формате Markdown с шаблонами и секциями «Резюме --- Метрики --- Инциденты --- Рекомендации».

6. Интегрировать с Telegram (библиотека python-telegram-bot версии 20.0+ для асинхронной работы), реализовать команды управления (/send_now, /set_report, /unset_report, /status) с механизмом авторизации по user_id и планирование отправки через JobQueue с использованием асинхронных задач для длительных операций сбора данных.

7. Провести тестирование (юнит и интеграционные тесты с использованием pytest и unittest.mock для имитации источников данных), документировать процесс развёртывания и использовать настраиваемые методы анонимизации для защиты персональных данных.

Защищаемые свойства информации.

Разрабатываемый проект способствует обеспечению следующих ключевых свойств информационной безопасности:

· **Доступность:** Путем оперативного мониторинга сбоев сервисов и сетевых атак (например, DDoS), что позволяет сократить время простоя.

· **Целостность:** За счет анализа логов на предмет признаков несанкционированного изменения системных файлов, настроек или данных.

· **Конфиденциальность:** Косвенно, путем обнаружения в логах инцидентов, связанных с попытками несанкционированного доступа к данным, а также за счёт использования механизмов анонимизации в обрабатываемых отчётах.

1. Исследование проблемы

1.1 Описание сферы и проблемы в целом

1.1.1 Общая характеристика профиля «Информационная безопасность»

Профиль «Информационная безопасность» в контексте автоматизации процессов мониторинга безопасности поднимает важные вопросы о доступности и эффективности защиты информационных систем. В условиях постоянного роста киберугроз, организации нуждаются в инструментах, которые не только помогут выявлять уязвимости, но и обеспечат быстрое реагирование на инциденты. Информационная безопасность охватывает широкий спектр задач, включая:

- Мониторинг сетевой активности.
- Выявление уязвимостей в приложениях.
- Анализ инцидентов.
- Создание и отправка отчетов о состоянии безопасности.

В этом контексте применение Telegram-ботов для автоматизации сбора данных и формирования отчетов становится особенно актуальным.

1.1.2 Конкретизация: автоматизация отчетности как киберзащита

При автоматизации отчетности важно понимать, что это не просто механизм для упрощения задач. Это важный аспект защиты, который позволяет быстро собирать и анализировать информацию о состоянии безопасности. Многие организации до сих пор используют устаревшие методы, которые требуют значительных временных затрат и не всегда дают точные результаты. Менеджеры по ИТ-безопасности часто сталкиваются с проблемой недостатка информации для оперативного реагирования на инциденты.

Преимущества автоматизации отчетности через Telegram-бота включают:

- Скорость: возможность моментального получения актуальной информации о состоянии безопасности.
- Удобство: доступ к отчетам через привычный интерфейс мессенджера.
- Экономия ресурсов: сокращение времени, затрачиваемого на ручной сбор данных.

1.2 Описание проблемы

Проблема заключается в том, что большинство организаций, особенно небольших и средних, не имеют эффективных инструментов для автоматизации процессов мониторинга безопасности. Ручной сбор и анализ данных требуют много времени и усилий, что может привести к задержкам в реагировании на угрозы и инциденты.

Где возникают проблемы:

1. **Людской фактор**: ошибки, допущенные при ручном вводе данных или анализе, могут привести к пропуску важных инцидентов.

2. **Нехватка времени**: специалисты часто перегружены, и у них нет возможности уделить внимание всем аспектам безопасности.

3. **Отсутствие интеграции**: системы, которые уже используются в организациях, часто не имеют возможности интеграции, что делает их использование неэффективным.

В результате, необходима система, которая бы могла автоматически собирать данные и формировать отчеты, оставляя возможность пользователю сосредоточиться на более сложных задачах.

1.3 Актуальность проблемы

С учетом текущих реалий, количество кибератак возрастает с каждым годом, что делает защиту информации одной из главных задач для организаций. По данным статистики, малые и средние предприятия чаще становятся жертвами киберпреступности.

В условиях ограниченных ресурсов, многие из них не могут позволить себе дорогостоящие решения для мониторинга безопасности.

Проблемы, с которыми они сталкиваются:

1. **Недостаток автоматизации**: многие организации все еще используют ручные методы работы, что не позволяет обеспечить достаточный уровень безопасности.

2. **Невозможность быстрого реагирования**: задержки в получении отчетов о состоянии безопасности могут привести к критическим последствиям, особенно в условиях быстрого изменения киберугроз.

3. **Доступность информации**: автоматический бот, работающий в Telegram, позволяет быстро и удобно получать информацию о состоянии безопасности, что значительно упрощает работу специалистов.

Автоматизация отчетности не только снижает время реакции на инциденты, но и делает мониторинг безопасности доступным для широкого круга организаций, включая малый бизнес и образовательные учреждения.

1.4 Анализ аналогичных решений

На рынке существует несколько решений для автоматизации процессов мониторинга безопасности, однако большинство из них ориентировано на крупные компании и требует значительных финансовых затрат. Например:

1. **Платные решения**: такие как Splunk или ELK Stack, которые предлагают мощные инструменты для анализа данных, но их стоимость (ELK — opensource) и сложность внедрения часто недоступны для малых предприятий. Они требуют обширной инфраструктуры и ресурсов для настройки, что делает их нецелесообразными для небольших организаций. В отличие от рассмотренных коммерческих SIEM-решений (Splunk, ELK), данный проект предлагает низкопороговое решение с нулевой стоимостью лицензии, ориентированное на малый бизнес и образовательные учреждения. В отличие от разрозненных скриптов, проект представляет собой целостную систему с модульной архитектурой, базой

данных для истории и удобным каналом доставки. Ключевым отличительным преимуществом является глубокая интеграция с мессенджером Telegram, что обеспечивает мгновенное получение отчётов в привычной для пользователя среде, без необходимости мониторинга электронной почты или веб-панелей.

2. **Ограниченные скриптовые решения**: многие разработчики создают скрипты для конкретных задач, которые могут автоматизировать некоторые аспекты мониторинга, но они не предлагают целостного решения. Пользователи часто сталкиваются с проблемами интеграции таких скриптов в существующие системы, а также с отсутствием поддержки и документации.

3. **Отсутствие интеграции с мессенджерами**: большинство текущих решений не позволяют удобно получать уведомления и отчёты о состоянии безопасности. Стандартные инструменты предлагают отправку отчетов по электронной почте, что не всегда быстро и удобно для пользователей. Это также может привести к перегрузке почтовых ящиков и затруднению поиска нужной информации.

4. **Недостаток гибкости**: существующие системы часто не позволяют настраивать частоту и формат отчетов под конкретные нужды пользователя. Это может стать проблемой для организаций, чьи потребности в отчетности отличаются друг от друга.

5. **Требования к техническим навыкам**: большинство из этих решений требует наличие технических специалистов, способных установить и поддерживать их работу, что является дополнительной нагрузкой для малых предприятий.

Таким образом, на рынке недостаточно доступных, гибких и простых в использовании решений для автоматизации отчетности по безопасности, что делает разработку телеграм-бота для этой цели интересным и актуальным вариантом. Он позволяет собирать и предоставлять информацию непосредственно в чатах, сохраняя при этом простоту использования и доступность, что особенно важно для малых и средних организаций, стремящихся повысить уровень своей безопасности без лишних затрат и сложностей.

1.5 Аудитория проекта

Мой проект нацелен на широкий круг пользователей: малые и средние бизнесы, образовательные учреждения и все организации, нуждающиеся в эффективном мониторинге состояния безопасности. Это решение важно для шефов информационных технологий, системных администраторов и специалистов по безопасности, которые

отвечают за обеспечение безопасности информации и систем. Благодаря автоматизации, эти специалисты получают возможность сосредоточиться на более важных задачах, а не тратить время на рутинные действия, что делает решение особенно актуальным для сферы, где ресурсы и время имеют критическое значение.

В конечном счёте, проект не только улучшает степень мобильности и доступности отчетности о состоянии безопасности, но и создает более безопасную и информированную рабочую среду для всех пользователей.

2. РЕШЕНИЕ ПРОБЛЕМЫ

2.1 Формализация проблемы

Пусть S = {s1,…,sn} — набор источников данных безопасности (логи, сканеры, API). Для каждого источника si в момент времени t поступает поток событий Ei(t). Пусть P — набор правил фильтрации и приоритизации (пороги CVSS, уровни критичности, временные окна). Требуется автоматическая функция Report: (S, P, [t0,t1]) → R, где R — структурированный отчёт, содержащий:

- MK — набор ключевых метрик (число инцидентов, число критичных уязвимостей, средняя загрузка и т.д.);

- I — список инцидентов с полями (id, time, source, severity, description);

- V — список уязвимостей с полями (CVE, component, CVSS, status);

- Meta — метаданные (период, недоступные источники, применённые фильтры).

Требование корректности: для фиксированных (S,P,[t0,t1]) результат R должен быть детерминирован, содержать MK,I,V, и сохраняться в БД для последующего сравнения. Система должна сохранять работоспособность при частичной недоступности источников и обеспечивать гарантии доставки отчёта в Telegram (retry-политика).

2.2 Описание алгоритма решения

Простой модульный алгоритм:

1. Инициализация

1.1. Загрузка конфигурации: S, P, расписание T, chat_id.

1.2. Подготовка адаптеров для каждого si.

2. Сбор данных за период [t0,t1]

2.1. Асинхронно опросить каждый адаптер collect(si,[t0,t1]) с таймаутом.

2.2. Зафиксировать недоступные источники в Meta.

3. Нормализация

3.1. Привести события к единой структуре (timestamp, type, severity, details).

3.2. Удалить дубликаты и скорректировать временные метки.

3.3. Применить политики анонимизации (удаление или маскирование персональных и идентифицирующих данных) к полям details нормализованных событий.

4. Агрегация и приоритизация

4.1. Вычислить МК (суммы, средние, изменения относительно предыдущего периода).

4.2. Выявить потенциальные уязвимости основываясь на открытых базах данных. (NVD — National Vulnerability Database)

5. Генерация отчёта

5.1. Подставить данные в шаблон Markdown (сводка, таблицы, список инцидентов, рекомендации).

6. Отправка и подтверждение доставки

6.1. Поместить отчёт в очередь отправки и вызвать Telegram API.

6.2. При ошибке — повторить отправку по политике; при длительном сбое — уведомить администратора.

7. Архивация

7.1. Сохранить отчёт и агрегаты в БД (таблица reports) для истории и аналитики.

2.3 Описание существующих алгоритмов

1) Поллинг (Polling)

- Суть: периодический опрос источников за новые данные.
- Плюсы: простота реализации, нет необходимости в публичных endpoint.
- Минусы: задержка получения событий, нагрузка при частых опросах.
- Применимость: первичный метод для малого проекта.

2) Вебхуки (Webhook / Push)

- Суть: источник сам отправляет события на endpoint.
- Плюсы: низкая задержка, экономия ресурсов.
- Минусы: требует настройки источников и доступного URL; вопросы безопасности.
- Применимость: для интеграций с поддерживающими сервисами.

3) Нормализация и дедупликация

- Суть: приведение разных форматов логов к общей структуре и удаление повторов.

- Практика: маппинг полей и использование timestamp + hash для дедупликации.

- Применимость: обязательна для корректной агрегации.

4) Rule-based корреляция

- Суть: простые правила связывания событий (например: N неудачных входов за M минут → инцидент).

- Плюсы: прозрачность и лёгкость настройки.

- Минусы: требует вручную задаваемых правил, может давать ложные срабатывания.

- Применимость: базовый метод для выявления значимых инцидентов в проекте.

5) Агрегация метрик и простая детекция аномалий

- Суть: подсчёт сумм/средних и выявление аномалий порогами или z-score.

- Плюсы: лёгкая реализация, понятные результаты для отчёта.

- Минусы: не учитывает сложных зависимостей.

- Применимость: рекомендовано использовать в первой версии.

6) Выявление уязвимостей

- Суть: Выявление известных уязвимостей в мониторируемых системах для включения в отчёт. (будет использоваться поиск с использованием открытой базы данных уявимостей ли интеграция с opensource сканером уязвимостей)

- Применимость: реализуется в простом виде для отчёта.

7) Шаблонизация отчётов

- Суть: генерация Markdown-отчётов через шаблоны (Jinja2).

- Плюсы: удобство чтения в Telegram и лёгкая кастомизация.

- Применимость: основной способ формирования вывода.

3. ОПИСАНИЕ РЕАЛИЗАЦИИ

3.1 Техническое задание

3.1.1 Цели и задачи

Цель: Разработать Telegram-бота для автоматизации отчетности по информационной безопасности, который будет собирать данные о состоянии кибербезопасности и генерировать отчеты.

Задачи:

1. Спроектировать архитектуру бота с учетом требований к функционалу.
2. Реализовать методы сбора данных о сетевой активности и инцидентах.
3. Создать механизм генерации отчетов в текстовом или Markdown-формате.
4. Настроить автоматизированную отправку отчетов в заданное время.
5. Обеспечить возможность настройки бота для пользователей.
6. Настроить постоянную систему мониторинга происшествий и немедленного оповещения пользователя в случае последних.

3.1.2 Требования

1. Язык программирования: Python 3.9+.
2. Библиотека: python-telegram-bot 20.0+ для работы с Telegram API.
3. Система хранения данных: SQLite (один файл для базы данных).
4. Код должен соответствовать стандартам ГОСТ и иметь документированное оформление.
5. Ведение логов всех операций с указанием даты и времени (и их своевременное удаление для предотвращения переполнения диска).
6. Реализация механизма авторизации пользователей на основе user_id Telegram для выполнения администраторских команд (например, /set_report).
7. Хранение конфиденциальных данных (токен бота, ключи API) в защищенных конфигурационных файлах (например, с использованием библиотеки python-dotenv) или переменных окружения.

8. Обеспечение целостности и доступности данных бота: настройка прав доступа к файлам базы данных и логов, регулярное резервное копирование критических данных (база отчетов, конфигурация).

3.1.3 Сценарии использования

1. Настройка: Пользователь (авторизованный администратор) настраивает частоту отчетов и желаемый контент через команды бота.

2. Сбор данных: Бот собирает информацию о состоянии безопасности по расписанию, обращаясь к внешним мониторинговым системам.

3. Генерация отчетов: Создание и отправка отчетов в текстовом или Markdown-формате пользователям в заданное время.

3.2 Описание идеи проекта

3.2.1 Краткая суть

Мы создаем Telegram-бота, который помогает собирать и анализировать данные о состоянии информационной безопасности. Это решение обеспечивает автоматизацию процессов мониторинга и отчетности, что значительно сокращает временные затраты специалистов.

3.2.2 Соответствие задачам олимпиады

При защите проекта будут представлены:

1. Рабочий бот, способный собирать и отправлять отчеты.

2. Пояснительная записка, в которой подробно изложены все этапы разработки.

3. Презентация с графиками, диаграммами и пользовательскими сценариями работы.

3.3 Используемые технологии и обоснование их использования

1. Python: Универсальный язык, который прост для изучения и активно используется в области разработки ботов. Позволяет легко интегрировать внешние API.

2. Библиотека python-telegram-bot: Предоставляет все необходимые инструменты для работы с Telegram API, что позволяет легко управлять сообщениями и событиями.

3. SQLite: Легкая система управления базами данных, позволяющая хранить данные в одном файле, что делает проект простым в развертывании. Архитектура проекта допускает замену на другую СУБД (например, PostgreSQL) при необходимости масштабирования.

4. Git: Используется для контроля версий проекта, что упрощает управление изменениями в коде и позволяет эффективно работать в команде.

5. Документация: Все материалы, включая код и пояснительные заметки, будут четко структурированы и задокументированы, чтобы облегчить понимание проекта.

6. pytest и unittest.mock: Используются для модульного и интеграционного тестирования, включая тестирование обработки ошибок и имитацию источников данных.

3.4 Принцип работы

3.4.1 Внешний вид и маршрут пользователя

Пользователь взаимодействует с ботом через Telegram-интерфейс. Бот поддерживает три ключевых сценария взаимодействия:

1. **Ручной запрос отчета:** Пользователь отправляет команду (например, /report_now или /status), чтобы немедленно получить отчет о текущем состоянии безопасности.

2. **Автоматические отчеты по расписанию:** Бот автоматически генерирует и отправляет регулярные отчеты в заданное время (например, ежедневно в 09:00) без участия пользователя.

3. **Экстренные уведомления:** При обнаружении критических инцидентов бот незамедлительно отправляет пользователю аварийный алерт, независимо от расписания или запросов.

Все взаимодействие происходит в чате Telegram, где пользователь получает структурированные сообщения от бота.

3.4.2 Сбор данных

Бот собирает информацию о состоянии безопасности из нескольких источников: системные логи, логи приложений, сетевые устройства, внешние API сканирования уязвимостей. Сбор данных организован в трех режимах:

· **По расписанию:** Регулярные проверки выполняются автоматически с заданным интервалом для формирования периодических отчетов.

· **По команде:** По запросу пользователя выполняется внеплановый сбор данных для немедленного отчета.

· **Непрерывный мониторинг:** Бот постоянно отслеживает ключевые метрики безопасности в реальном времени для выявления критических инцидентов. При превышении пороговых значений (например, 100 неудачных попыток входа за 5 минут) система мгновенно инициирует отправку алерта.

Для обеспечения отзывчивости бота длительные операции сбора выполняются асинхронно или в фоновых задачах, что не блокирует обработку команд и отправку уведомлений.

3.4.3 Генерация отчетов

На основе собранных данных бот формирует два типа сообщений:

1. Регулярные отчеты: Формируются по расписанию или по запросу в формате Markdown и включают ключевые метрики, список инцидентов, уязвимости и рекомендации за определенный период.

2. Экстренные алерты: Генерируются мгновенно при обнаружении критических событий. Алерт содержит краткое описание инцидента, время обнаружения и первоочередные рекомендации по реагированию.

3.4.4 Демонстрация проекта

На защите будет показан процесс взаимодействия с ботом, включая команды для настройки, автоматическое получение и отправку отчетов. Визуализация через пользовательский интерфейс Telegram позволит продемонстрировать функциональность и

удобство использования бота. Будет также продемонстрирован сценарий отказа в доступе к административной команде для неавторизованного пользователя.

3.5 Доказательство работоспособности

3.5.1 Практический эксперимент

1. Первичный вход:Пользователь настраивает бота через команды, вводя желаемую частоту отчетов.

- Логи фиксируют действия:
- REQUEST: пользователь отправляет команду на настройку.
- CREATE_SESSION: бот успешно создает сессию для пользователя.

2. Сбор данных: Бот автоматически инициирует запросы к внешним API для получения информации о состоянии безопасности.

- Логи фиксируют действия:
- DATA_COLLECTION: бот получает данные о сетевой активности и инцидентах. Все полученные данные сохраняются в базу данных.

3. Генерация отчетов: В установленное время бот создает отчет на основе собранной информации.

- Логи фиксируют действия:
- REPORT_GENERATION: бот успешно генерирует отчет в заданном формате.
- REPORT_DISPATCH: бот отправляет отчет пользователю в Telegram.

4. Автоматизированная отправка: В ходе тестирования бот регулярно отправляет отчеты по расписанию.

- В результате проверок подтверждается, что все отправленные отчеты содержат актуальную и корректную информацию о состоянии безопасности.

3.5.2 Соответствие требованиям

Все аспекты проекта соответствуют заявленным требованиям, включая:

1. Технические аспекты:

- Использование Python 3.9+ и библиотеки python-telegram-bot 20.0+ обеспечивают высокую стабильность и надежность работы бота.

- SQLite позволяет легко хранить и управлять данными без необходимости настройки сложной серверной структуры. Файл базы данных защищен правами доступа операционной системы.

2. Документация:

- Каждый этап разработки задокументирован, включая код, инструкции и логи. Это облегчает понимание и позволяет при необходимости вносить изменения.

3. Соблюдение стандартов:

- Проект оформлен в соответствии с требованиями ГОСТ. Все документы структурированы и содержат необходимые разделы, как и требуется для участия в олимпиаде.

4. Аспекты безопасности:</ins>

-Реализована авторизация административных команд.

-Конфиденциальные данные хранятся отдельно от кода.

-Применяются механизмы анонимизации в обрабатываемых данных.

-Обеспечены меры по защите целостности и доступности данных бота (права доступа, бэкапы).

3.6 Перспективы развития проекта

В перспективе проект может быть значительно расширен:

1. Добавление новых функций:

- Возможность интеграции с различными API для получения дополнительной информации по киберугрозам и уязвимостям.

- Расширение функционала бота за счет введения новых команд для более детального анализа и обработки данных.

- Реализация более сложных методов анонимизации и шифрования конфиденциальных данных в отчетах.

2. Интерфейс управления:

- Разработка веб-интерфейса для управления ботом, настройки периодичности отчетов и выбора целевой аудитории.

3. Внедрение машинного обучения:

- Использование алгоритмов машинного обучения для более глубокой аналитики и предсказания инцидентов на основе исторических данных.

4. Расширение аудитории:

- Создание версии бота для различных мессенджеров, таких как WhatsApp и Viber, для привлечения более широкой аудитории.

5. Коммерческое применение:

- При должном тестировании и доработке проект может стать коммерчески жизнеспособным продуктом, который будет использоваться компаниями для мониторинга своей кибербезопасности.

Это позволит не только улучшить функционал, но и сделать проект полезным инструментом для пользователей в сфере информационной безопасности.

ЗАКЛЮЧЕНИЕ

В данной пояснительной записке представлены основные моменты разработки и внедрения проекта «Telegram-бот для автоматизации отчетности по информационной безопасности». Проект направлен на упрощение процессов мониторинга и отчетности в области кибербезопасности, что стало актуальным в условиях растущих угроз.

Достигнутые результаты (положительные стороны проекта):

1. Разработан рабочий прототип Telegram-бота на Python, интегрированный с библиотекой python-telegram-bot и базой данных SQLite.
2. Реализована модульная архитектура (Collector, Normalizer, Aggregator, ReportGenerator), соответствующая поставленным задачам.
3. Спроектирован и описан детальный алгоритм сбора, нормализации, агрегации данных и генерации отчётов.
4. Проект решает актуальную проблему автоматизации рутинной отчётности для малых организаций, предлагая простое и доступное решение.
5. Проведено тестирование работоспособности основных сценариев использования с использованием инструментов pytest и unittest.mock

Перспективы развития

Разработанный бот имеет большой потенциал для дальнейшего расширения функциональности:

- Интеграция с другими мессенджерами. Рассмотрение возможности интеграции с такими платформами, как WhatsApp или Viber, для расширения аудитории.
- Машинное обучение. Использование технологий машинного обучения для более точного предсказания инцидентов и автоматизации реакций на угрозы.
- Веб-интерфейс. Создание веб-интерфейса для удобного управления и мониторинга бота, что упростит взаимодействие конечных пользователей с системой.
- Расширенная аналитика. Внедрение инструментов для глубокой аналитики данных, что позволит повысить качество и точность отчетов.

Недоработки и ограничения текущей версии:

1. Бот функционирует исключительно в экосистеме Telegram, что сужает потенциальную аудиторию.

2. Анализ данных основан на rule-based подходах, отсутствуют элементы машинного обучения для продвинутой аналитики и предсказания инцидентов.

3. Масштабируемость проекта может быть ограничена использованием SQLite при очень высокой нагрузке; однако архитектура позволяет заменить СУБД на более производительную (например, PostgreSQL)

4. Требуется доработка пользовательского интерфейса (UI) для более гибкой настройки без прямого редактирования конфигурационных файлов.

5. Требуется усиление мер безопасности для защиты самого бота и хранимых данных (регулярное обновление зависимостей, мониторинг уязвимостей в используемых библиотеках).

Итог

Таким образом, проект «Бот для автоматизации отчетности по информационной безопасности в Telegram» отвечает на текущие требования мониторинга безопасности и открывает новые возможности для исследований и доработок. Успешная реализация системы может стать основой для дальнейших инициатив в сфере кибербезопасности и улучшения защиты информационных систем.

Рекомендации

Для повышения степени защиты и удобства пользователей рекомендуются следующие дополнительные функции:

- Добавление режима отладки. Возможность включения детального логирования для упрощения диагностики проблем.

- Улучшение пользовательского интерфейса. Разработка более интуитивного интерфейса для пользователей, что повысит их взаимодействие с ботом.

- Обратная связь от пользователей. Внедрение механизма сбора отзывов от пользователей для непрерывного улучшения функциональности.

- Внедрить систему регулярного резервного копирования базы данных отчетов и логов бота.

- Регулярно обновлять зависимости (библиотеки) для устранения известных уязвимостей.

Проект имеет потенциал не только для решения текущих задач в области безопасности, но и для дальнейшего совершенствования и интеграции в более широкие системы защиты.

СПИСОК ЛИТЕРАТУРЫ

1. Официальная документация Python [Электронный ресурс]. – URL: https://docs.python.org/3/
2. Python Telegram Bot library documentation [Электронный ресурс]. – URL: https://docs.python-telegram-bot.org/
3. Национальный стандарт РФ ГОСТ Р ИСО/МЭК 27000-2012 "Информационная технология. Методы и средства обеспечения безопасности. Системы менеджмента информационной безопасности. Общие положения и словарь". – М.: Стандартинформ, 2012.
4. Шаньгин В.Ф. Информационная безопасность компьютерных систем и сетей: учебное пособие. – М.: ИД «Форум»: ИНФРА-М, 2012. – 416 с.
5. MITRE ATT&CK Framework [Электронный ресурс]. – URL: https://attack.mitre.org/
6. Общие критерии оценки безопасности информационных технологий (Common Criteria) [Электронный ресурс]. – URL: https://www.commoncriteriaportal.org/
7. Документация по SQLite [Электронный ресурс]. – URL: https://www.sqlite.org/docs.html
8. Документация по библиотеке python-dotenv \[Электронный ресурс\]. -- URL: https://pypi.org/project/python-dotenv/
9.Документация по pytest \[Электронный ресурс\]. -- URL: https://docs.pytest.org/