Автоматизированная система инвентаризации ПО и оценки критичности программных угроз «Vulnerability Scanner

Кадырканова Аруuке, ОИСП-292

Введение и актуальность

Цель – создать инструмент для автоматического обнаружения программ и сопоставления их с глобальными базами уязвимостей.

Задачи

- организация прямого доступа к системному реестру **Windows** для инвентаризации ПО
- проектирование базы данных на платформе **PostgreSQL**
- разработка механизмов асинхронного взаимодействия с внешним **API** сервиса **NVD.**

Функциональные возможности

- Сбор данных: Извлечение информации о версиях и производителях ПО из веток реестра **HKLM** и **HKCU**.
- Анализ угроз: Выполнение сетевых запросов к базе уязвимостей и расчет метрик по шкале **CVSS**.
- Интерфейс: Формирование аналитического отчета с цветовой маркировкой уровней риска.
- Управление данными: Фильтрация записей в реальном времени и безопасная очистка таблиц.

Технологический стек

- Язык программирования: **Python**
- Библиотеки интерфейса: **PyQt6**
- Работа с данными: **psycopg2** (**PostgreSQL**) и **winreg** (реестр **Windows**).
- Сетевое взаимодействие: Библиотека **requests** для работы с **API**.
- Архитектурный паттерн: **MVC** (**Model-View-Controller**) и паттерн **Observer**.

Архитектура и программные компоненты

Метод сбора данных из системного реестра

```
def get_installed_software():
    software_list = []

    paths = [
        (winreg.HKEY_LOCAL_MACHINE, r"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall"),
        (winreg.HKEY_LOCAL_MACHINE, r"SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall"),
        (winreg.HKEY_CURRENT_USER, r"SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall")
    ]

    for root, path in paths:
        try:
            reg_key = winreg.OpenKey(root, path)
            num_subkeys = winreg.QueryInfoKey(reg_key)[0]
```

Архитектура и программные компоненты

Реализация многопоточности (QThread)

```
class UpdateWorker(QThread):
    progress = pyqtSignal(str)
    finished = pyqtSignal()

    def run(self):
        headers = {"apiKey": api_key}
        try:
            conn = psycopg2.connect(**db_params)
            cursor = conn.cursor()

            cursor.execute("select distinct display_name from software_inventory")
            software_list = cursor.fetchall()

            total = len(software_list)
            for index, (name,) in enumerate(software_list):
                words = name.split()
                search_term = f"{words[0]} {words[1]}" if len(words) > 1 else name
                encoded_term = urllib.parse.quote(search_term)

                self.progress.emit(f"[{index+1}/{total}] Поиск: {search_term}...")
                time.sleep(0.6)

                url = f"https://services.nvd.nist.gov/rest/json/cves/2.0?keywordSearch={encoded_term}&resultsPerPage=20"
```

Архитектура и программные компоненты

Интеграция с базой данных и визуализация рисков

```python
def load_dashboard_data():
    try:
        conn = psycopg2.connect(**db_params)
        cursor = conn.cursor()

        cursor.execute("select soft_name, version, cve_id, risk_level, risk_score from security_dashboard_v")
        rows = cursor.fetchall()

        form.table_dashboard.setRowCount(len(rows))
        form.table_dashboard.setColumnCount(5)
        form.table_dashboard.setHorizontalHeaderLabels(["ПО", "Версия", "CVE ID", "Уровень", "Score"])
        form.table_dashboard.horizontalHeader().setSectionResizeMode(QHeaderView.ResizeMode.Stretch)

        for i, (name, ver, cve, level, score) in enumerate(rows):
            form.table_dashboard.setItem(i, 0, QTableWidgetItem(str(name)))
            form.table_dashboard.setItem(i, 1, QTableWidgetItem(str(ver)))
            form.table_dashboard.setItem(i, 2, QTableWidgetItem(str(cve)))

            level_item = QTableWidgetItem(str(level))
            if level == 'critical':
                level_item.setBackground(QColor(255, 100, 100))
```

Спасибо!