# ЭТАПЫ РАЗРАБОТКИ ИНСТРУМЕНТА ДЛЯ ВЫЯВЛЕНИЯ УЯЗВИМОСТЕЙ В СОВРЕМЕННЫХ ОС С ОТКРЫТЫМ КОДОМ

***О.О. Турсунов, С.М. Бозоров***

*Ташкентский университет информационных технологий имени Мухаммада ал-Хоразмий*

**Аннотация:** Статья описывает этапы разработки инструмента для выявления уязвимостей в современных операционных системах с открытым кодом. Авторы подробно рассматривают процесс подготовки к проектированию инструмента, анализ требований, определение функциональности, проектирование архитектуры, разработку прототипа, проверку обновлений ПО, проверку безопасности настроек и использования, а также доработку и оптимизацию функционала инструмента.

**Ключевые слова:** Уязвимости, операционные системы с открытым кодом, разработка инструмента, анализ требований, функциональность, архитектура, прототип, обновление по, безопасность настроек, оптимизация функционала.

*Подготовка к проектированию инструмента. Анализ требований и определение функциональности*

Перед тем, как взяться за разработку новейшего инструмента для выявления уязвимостей в операционных системах с открытым кодом, необходима тщательная подготовка. Этот начальный этап играет ключевую роль в определении последующего успеха проекта. Важно провести анализ требований конкретной категории пользователей, выявить основные кейсы использования будущего продукта, и, конечно же, четко сформулировать функциональные возможности, которые должны быть доступны в данном инструменте.

В этом разделе мы определяем основные требования к нашему сканеру безопасности для ОС Linux и выявляем функциональность, необходимую для достижения этих целей.

Наш сканер должен выполнять такие функции:

– обнаружение уязвимостей через программные обновления;

– анализ конфигурации аутентификации для поиска уязвимостей;

– поиск уязвимостей в ядре операционной;

– оценка наличия антивирусных и анти-малваре угроз;

– анализ открытых портов и сервисов для выявления уязвимостей;

– проверка файрвола на предмет уязвимостей;

– исследование IDS/IPS и инструментов резервного копирования для обнаружения уязвимостей;

– проверка прав доступа на предмет уязвимостей.

Основной функционал нашего сканера будет включать в себя сканирование уязвимостей, анализ конфигурации системы, генерацию отчетов и возможность интеграции с другими инструментами безопасности.

*Проектирование архитектуры инструмента для проверки безопасности ОС*

На этом этапе мы определяем архитектуру нашего сканера безопасности для ОС Linux, чтобы обеспечить эффективную и расширяемую реализацию.

Мы выбираем модульную архитектуру для нашего сканера, чтобы разделить функциональность на отдельные компоненты и обеспечить их независимое развитие и тестирование.

*Разработка прототипа инструмента. Создание базового функционала для тестирования и дальнейшего улучшения*

На этом этапе мы создаем прототип нашего сканера безопасности для ОС Linux, чтобы протестировать его базовую функциональность.

```
def check_updates():
    cache = apt.Cache()

    # Создаем список для хранения необновленных программ и их доступных обновлений
    outdated_packages = []

    # Проходимся каждой программе на наличие обновлений
    for package in cache:
        pkg = cache[package]

        # Проверяем, есть ли доступные обновления для программы
        if pkg.is_upgradable:
            outdated_packages.append((pkg.name, pkg.installed.version, pkg.candidate.version))

    return outdated_packages
```

Рис. 1 - Код для проверки обновлений программного обеспечения

*Проверка обновлений программного обеспечения в ОС Linux*

Этот код предназначен для автоматизации проверки наличия обновлений программного обеспечения в Linux и сохранения результатов этой проверки в текстовый файл.

*Проверка и сбор информации о ядре Linux*

```
def check_kernel_version():
    kernel_version = os.uname().release
    if kernel_version:
        print("\033[1;33m[+] Kernel\033[0m")
        print("------------------------------------")
        print("Checking kernel version [ DONE ]")
        return kernel_version
    else:
        print("\033[1;33m[+] Kernel\033[0m")
        print("------------------------------------")
        print("Checking kernel version [ UNKNOWN]")
        return None
```

Рис. 2 - Методы проверки версии, выпуска и типа ядра Linux

```
def check_loaded_kernel_modules():
    loaded_modules = os.listdir('/sys/module')
    if loaded_modules:
        print("Checking loaded kernel modules [ DONE ]")
        print("Found", len(loaded_modules), "active modules")
        return loaded_modules
    else:
        print("Checking loaded kernel modules [ UNKNOWN] ")
        return None

2 usages
def check_kernel_config_file():
    config_file = '/proc/config.gz'
    if os.path.exists(config_file):
        print("Checking Linux kernel configuration file [ FOUND ]")
        return True
    else:
        print("Checking Linux kernel configuration file [ NOT FOUND ]")
        return False

2 usages
def check_available_kernel_update():
    cache = apt.Cache()
    cache.open()
    cache.upgrade()

    kernel_updates = [pkg for pkg in cache.get_changes() if 'linux-image' in pkg.name]

    if kernel_updates:
        print("Checking for available kernel update  [ AVAILABLE ]")
        print("\n")
        for update in kernel_updates:
            print(update.name)
        return kernel_updates
    else:
        print("Checking for available kernel update  [ OK ]")
        print("\n")
        return None
```

Рис. 3 - Код для проверки загруженных модулей ядра, наличия файла конфигурации ядра и доступных обновлений ядра

Эти коды собирают информацию о текущем состоянии ядра Linux, включая его версию, релиз, тип, загруженные модули, наличие конфигурационного файла и доступность обновлений.

```
def find_intrusion_detection_tools():
    intrusion_detection_tools = []
    # Пути для поиска инструментов обнаружения и предотвращения вторжений
    snort_path = "/etc/snort"
    suricata_path = "/etc/suricata"
    ossec_path = "/var/ossec"
    # Поиск Snort
    if directory_exists(snort_path):
        intrusion_detection_tools.append("Snort")
    # Поиск Suricata
    if directory_exists(suricata_path):
        intrusion_detection_tools.append("Suricata")
    # Поиск OSSEC
    if directory_exists(ossec_path):
        intrusion_detection_tools.append("OSSEC")
    return intrusion_detection_tools
```

Рис. 4 - Код для поиска инструментов обнаружения и предотвращения вторжений (IDS/IPS)

Эти коды проверяют наличие системных инструментов, связанных с автоматизацией, обнаружением вторжений и резервным копированием. Они ищут определенные программы и каталоги в системе.

*Проверка настройки и использования файрвола с помощью iptables*

```
    try:
        subprocess.run( args: ["iptables", "-L"], check=True, capture_output=True, text=True)
        print("  - Checking firewalls                    [ FOUND ]")
        print("\n")
    except subprocess.CalledProcessError:
        print("  - Checking firewalls                    [ NOT FOUND ]")
        print("\n")
```

Рис. 5 - Код для проверки наличия файрвола

```
def check_iptables_policies():
    try:
        # Проверка политик iptables для цепей filter
        result = subprocess.run( args: ["iptables", "-t", "filter", "-L", "--numeric"], check=True, capture_output=True,
                                text=True)
        output_lines = result.stdout.split('\n')
        policies = []
        for line in output_lines:
            if line.startswith("Chain"):
                chain_name = line.split()[1]
                policy = line.split()[-1]
                policies.append(f"Chain {chain_name}: Policy {policy}")

        return policies

    except subprocess.CalledProcessError as e:
        print("Ошибка при выполнении команды iptables:", e)
```

Рис. 6 -Код для проверки политик iptables для цепей filter

*Проверка безопасности и рекомендации по усилению*


Рис. 7 - Код для проверки наличия установленных антивирусных и анти-малварных программ

Эти коды проверяют наличие антивирусных программ и компиляторов в системе Linux, а также рекомендует меры по укреплению безопасности на основе результатов проверки.

*Проверка безопасности настроек аутентификации*

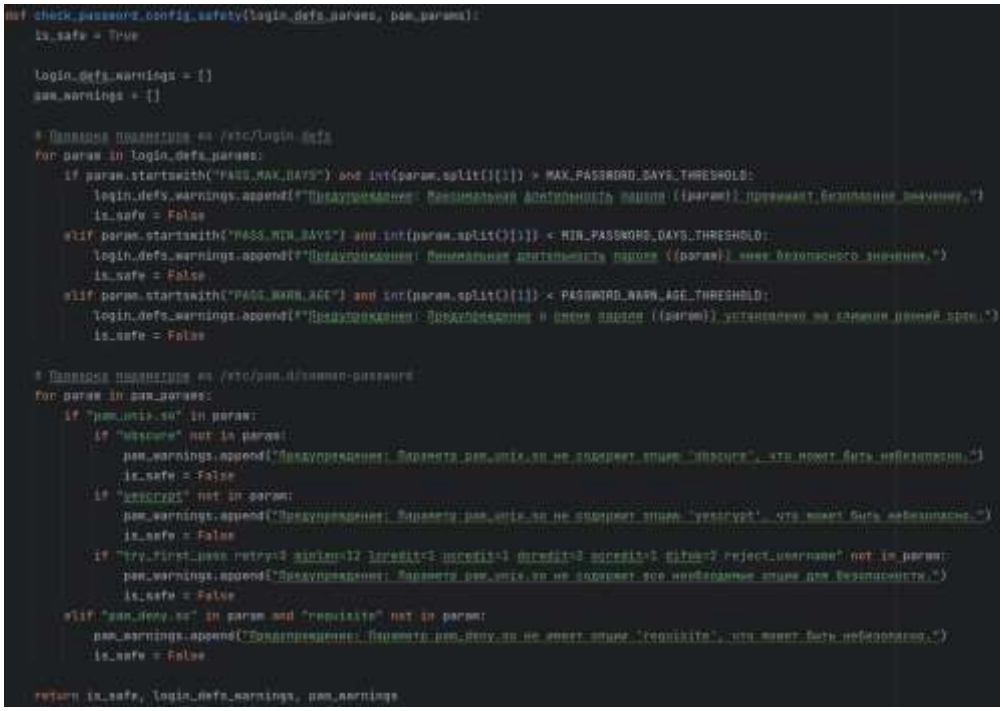

Рис. 8 - Код для проверки безопасности конфигурации паролей в файлах /etc/login.defs и /etc/pam.d/common-password

Этот код проверяет настройки безопасности аутентификации, а именно параметры пароля из файлов конфигурации /etc/login.defs и /etc/pam.d/common-password. Он ищет определенные параметры пароля и выдает предупреждения, если они находятся за пределами безопасных значений.

*Внесение необходимых корректив в инструмент на основе результатов тестирования*

После запуска скриптов выводило очень много информации. Это не удобно. Для того чтобы выводило нужная информация в терминале я доработал код так чтобы выводило только нужная информация в терминале, а остальное записывается в текстовый файл для подробного анализа.

Я также внёс изменения в код, чтобы добавить разноцветные выделения, позволяющие легче различать заголовки от остального результата.

*Доработка и оптимизация функционала инструмента*

Приступив к финальной стадии в работе над инструментом для выявления уязвимостей в современных операционных системах с открытым кодом, наступает время для доработки и оптимизации функционала.

Я создал файл main_script.py, чтобы избежать вызова всех функций по отдельности.

Теперь можно выполнить все функции сканирования одной командой.

Заключение

Следует отметить, что современные операционные системы становятся все более сложными и функциональными, что предоставляет больше возможностей для появления уязвимостей различного характера. Большое количество служб, модулей и приложений в современных ОС расширяет атакуемую поверхность, делая систему более уязвимой к внешним угрозам. Кроме того, постоянное обновление операционных систем не всегда гарантирует исправление всех обнаруженных уязвимостей, что оставляет двери для потенциальных атак. В современном информационном обществе, где данные становятся все ценнее и киберпреступники все более выверенными в своих действиях, вопрос обеспечения безопасности операционных систем становится критическим. Утечка или

несанкционированный доступ к данным может привести к серьезным финансовым и репутационным потерям для организации, а также поставить под угрозу личную безопасность конечного пользователя. Этот инструмент сканирования поможет специалистам по кибербезопасности и системным администраторам предотвратить угрозы для современных операционных систем с открытым кодом.

Список использованной литературы

1. Гарсия А. и Ли С. (2020). Повышение безопасности данных в операционных системах с открытым исходным кодом.

2. Сидоров К. Н. Защита компьютерной среды от киберугроз. - Киев: Издательство Украинская кибернетика, 2017.

3. Тарасов Г. Р. Комплексный подход к обеспечению безопасности информационных систем. - СПб: Наука и Техника, 2019.

4. Хакерская А., Пронин Г. «Инструменты проверки безопасности операционных систем». Международная конференция по кибербезопасности, 2019.