

Министерство образования и науки Российской Федерации  
Санкт-Петербургский Политехнический Университет Петра Великого

—  
Институт прикладной математики и механики  
**Кафедра «Информационная безопасность компьютерных систем»**

## **ЛАБОРАТОРНАЯ РАБОТА № 1**

**«Изучение средств анализа исходного кода»**

**по дисциплине «Безопасность операционных систем»**

Выполнил  
студент гр. 43609/1

<подпись>

Куликов Д.А.

Преподаватель

<подпись>

Жуковский Е.В.

Санкт-Петербург  
2019

## **1 Цель работы**

Изучение типовых уязвимостей и средств поиска уязвимостей на основе анализа исходного кода программного обеспечения.

## **2 Формулировка задания**

В ходе выполнения лабораторной работы необходимо выполнить следующие действия:

1. Изучить наиболее характерные типы уязвимостей для указанного языка программирования. Составить перечень данных типов уязвимостей с указанием описания и примерами уязвимого кода.

2. Сформировать выборку из 3 проектов для дальнейшей проверки эффективности работы сканеров уязвимостей исходного кода. 2 проекта должны заведомо содержать фрагменты уязвимого кода различного типа (тестовые проекты). 1 проект должен являться проектом с открытым исходным кодом (например, с [github.com](https://github.com)), объемом не менее 20 тысяч строк кода.

3. Описать выбранные приложения, их назначение, принцип работы и присутствующие в них уязвимости.

4. Классифицировать присутствующие в коде уязвимости по типу ошибки (CWE, Common Weakness Enumeration).

5. Изучить существующие средства анализа исходного кода (сканеры уязвимостей) для указанного языка программирования и выбрать для дальнейшего тестирования 3 средства.

6. Описать принципы работы и особенности выбранных сканеров уязвимостей.

7. Провести тестирование выбранных трех сканеров уязвимостей на трех выбранных приложениях.

8. Проанализировать всю информацию, представленную сканерами. Провести проверку правильности решений средства анализа о наличие ошибок / уязвимостей в приложении.

9. Составить сводную таблицу с результатами анализа уязвимых приложений, с указанием перечня всех присутствующих в приложении уязвимостей.

10. Сделать выводы об эффективности выбранных средств анализа в выявлении уязвимостей различных типов

### 3 Ход работы

CWE-15: Внешнее управление системой или настройкой конфигурации. Пользователь может осуществлять внешний контроль над одной или несколькими настройками системы или элементами конфигурации.

```
/* POTENTIAL FLAW: set the hostname to data obtained from a potentially external source */
if (!SetComputerNameA(data))
{
    printLine("Failure setting computer name");
    exit(1);
}
```

CWE-23: Относительный обход пути - программное обеспечение использует внешний ввод для создания имени пути, которое должно быть в ограниченном каталоге, но оно не нейтрализует должным образом последовательности, такие как «..», которые могут преобразовываться в местоположение, находящееся за пределами этого каталога.

```
data = dataBuffer;
/* FIX: Use a fixed file name */
strcat(data, "file.txt");
{
    FILE *pFile = NULL;
    /* POTENTIAL FLAW: Possibly opening a file without validating the file name or path */
    pFile = FOPEN(data, "wb+");
    if (pFile != NULL)
    {
        fclose(pFile);
    }
}
```

CWE-36: Абсолютный обход пути. Программное обеспечение использует внешний ввод для создания имени пути, которое должно находиться в ограниченном каталоге, но оно не корректно нейтрализует последовательности абсолютного пути, такие как «/ abs / путь», которые могут преобразовываться в местоположение, которое является за пределами этого каталога.

```
#ifdef _WIN32
/* FIX: Use a fixed, full path and file name */
strcat(data, "c:\\temp\\file.txt");
#else
/* FIX: Use a fixed, full path and file name */
strcat(data, "/tmp/file.txt");
#endif
{
    FILE *pFile = NULL;
    /* POTENTIAL FLAW: Possibly opening a file without validating the file name or path */
    pFile = FOPEN(data, "wb+");
    if (pFile != NULL)
    {
        fclose(pFile);
    }
}
```

CWE-78: Неправильная нейтрализация специальных элементов, используемых в команде ОС («Внедрение команды ОС»). Программное обеспечение создает всю или часть команды ОС, используя входные данные из внешнего компонента.

```
/* POTENTIAL FLAW: Execute command without validating input possibly leading to command injection */
EXECL(COMMAND_INT_PATH, COMMAND_INT_PATH, COMMAND_ARG1, COMMAND_ARG3,
NULL);
```

CWE-90: Неправильная нейтрализация специальных элементов, используемых в запросе LDAP («Инъекция LDAP»). Программное обеспечение создает весь или часть запроса LDAP с использованием внешнего воздействия входного компонента, но не нейтрализует или

неправильно нейтрализует специальные элементы, которые могут изменить предполагаемый запрос LDAP при его отправке в нижестоящий компонент.

CWE-114: Управление процессом - Выполнение команд или загрузка библиотек из ненадежного источника или в ненадежной среде может привести к тому, что приложение выполнит вредоносные команды (и полезные нагрузки) от имени злоумышленника.

```
HMODULE hModule;  
/* POTENTIAL FLAW: If the path to the library is not specified, an attacker may be able to  
 * replace his own file with the intended library */  
hModule = LoadLibraryA(data);
```

CWE-121: Переполнение буфера на основе стека. Условие переполнения буфера на основе стека - это условие, при котором перезаписываемый буфер выделяется в стеке (т. е. Является локальной переменной или, редко, параметром для функции).

```
/* FLAW: Use the sizeof(structCharVoid) which will overwrite the pointer voidSecond */  
memcpy(structCharVoid.charFirst, SRC_STR, sizeof(structCharVoid));
```

CWE-122: Переполнение буфера на основе кучи. Условие переполнения кучи - это переполнение буфера, при котором буфер, который может быть перезаписан, размещается в части кучи памяти, обычно это означает, что буфер был выделен с помощью процедуры, такой как malloc ().

```
charVoid * structCharVoid = (charVoid *)malloc(sizeof(charVoid));  
if (structCharVoid == NULL) {exit(-1);}  
structCharVoid->voidSecond = (void *)SRC_STR;  
/* Print the initial block pointed to by structCharVoid->voidSecond */  
printLine((char *)structCharVoid->voidSecond);  
/* FLAW: Use the sizeof(*structCharVoid) which will overwrite the pointer y */  
memcpy(structCharVoid->charFirst, SRC_STR, sizeof(*structCharVoid));
```

CWE-123: Условие запись-что-где - любое условие, при котором злоумышленник может записать произвольное значение в произвольное местоположение, часто в результате переполнения буфера.

CWE-124: Underffrite Buffer ('Buffer Underflow') - программное обеспечение выполняет запись в буфер, используя индекс или указатель, который ссылается на ячейку памяти до начала буфера.

CWE-126: Перечитывание буфера - программа считывает данные из буфера, используя механизмы доступа к буферу, такие как индексы или указатели, которые ссылаются на ячейки памяти после целевого буфера.

CWE-127: Недостаточное чтение буфера - программа считывает данные из буфера, используя механизмы доступа к буферу, такие как индексы или указатели, которые ссылаются на ячейки памяти до целевого буфера.

CWE-134: Использование строки формата с внешним управлением - Программное обеспечение использует функцию, которая принимает строку формата в качестве аргумента, но строка формата исходит из внешнего источника.

```
/* POTENTIAL FLAW: Do not specify the format allowing a possible format string vulnerability */  
fprintf(stdout, data);
```

CWE-176: неправильная обработка кодировки Unicode - программа неправильно обрабатывает, когда вход содержит кодировку Unicode.

CWE-188: опора на структуру данных / памяти - программное обеспечение делает неверные предположения о том, как данные протокола или память организованы на более низком уровне, что приводит к непреднамеренному поведению программы.

```
/* FLAW: Attempt to modify intSecond assuming intSecond comes after charFirst and  
 * is aligned on an int-boundary after charFirst */  
*(int*)(charPtr + sizeof(int)) = 5;
```

CWE-190: Целочисленное переполнение - программа выполняет вычисление, которое может привести к целочисленному переполнению, когда логика предполагает, что результирующее значение всегда будет больше исходного значения. Это может привести к другим недостаткам, когда расчет используется для управления ресурсами или для контроля.

CWE-191: Integer Underflow (Wrap или Wraparound) - продукт вычитает одно значение из другого, так что результат меньше минимально допустимого целочисленного значения, что приводит к значению, которое не равно правильному результату.

CWE-194: Неожиданное расширение знака - программное обеспечение выполняет операцию с числом, которое заставляет его расширять знак при преобразовании в больший тип данных. Когда исходное число отрицательно, это может привести к неожиданным значениям, которые приводят к уязвимости.

```
/* POTENTIAL FLAW: malloc() takes a size_t (unsigned int) as input and therefore if it is negative,
 * the conversion will cause malloc() to allocate a very large amount of data or fail */
char * dataBuffer = (char *)malloc(data);
```

CWE-195: Ошибка преобразования со знаком в unsigned. Программное обеспечение использует signed тип и выполняет приведение к unsigned примитиву, что может привести к неожиданному значению, если значение signed примитива не может быть представлено с использованием unsigned примитива.

CWE-196: Ошибка преобразования без знака в число со знаком. Программное обеспечение использует unsigned тип и выполняет приведение к signed, что может привести к неожиданному значению, если значение unsigned примитива не может быть представлено с использованием signed.

CWE-197: Ошибка числового усечения. Ошибки усечения возникают, когда тип приводится к типу меньшего размера и данные теряются при преобразовании.

```
/* POTENTIAL FLAW: Convert data to a char, possibly causing a truncation error */
char charData = (char)data;
printHexCharLine(charData);
```

CWE-222: усечение информации, относящейся к безопасности. Приложение усекает отображение, запись или обработку информации, относящейся к безопасности, таким образом, чтобы можно было скрыть источник или природу атаки.

```
username[USERNAME_SIZE] = '\0';
/* INCIDENTAL CWE 188 - reliance on data memory layout
 * recv and friends return "number of bytes" received
 * fwrite wants "the size of". ANSI/ISO allows the size of chars
 * to be anything (32 bits, 9 bits, etc.) so technically you
 * have to do conversion between these values
 */
/* FLAW: username is truncated before being logged */
memcpy(truncatedUsername, username, sizeof(truncatedUsername));
```

CWE-223: Пропуск информации, относящейся к безопасности. Приложение не записывает и не отображает информацию, которая была бы важна для определения источника или характера атаки или определения того, является ли действие безопасным.

```
/* FLAW: username is not logged */
fprintf(stderr, "Attempted login\n");
```

CWE-226: конфиденциальная информация не очищена перед выпуском - программное обеспечение не полностью очищает ранее использованную информацию в структуре данных, файле или другом ресурсе, прежде чем сделать этот ресурс доступным для стороны в другой сфере управления.

CWE-242: Использование по своей природе опасной функции - Программа вызывает функцию, которая не может гарантировать безопасную работу.

CWE-244: Неправильная очистка памяти кучи перед выпуском («Проверка кучи») - Использование функции realloc () для изменения размера буферов, в которых хранится конфиденциальная информация, может оставить уязвимую информацию уязвимой для атаки, поскольку она не удаляется из памяти.

```
/* FLAW: free() password without clearing the password buffer */
free(password);
```

CWE-252: непроверенное возвращаемое значение -

Программное обеспечение не проверяет возвращаемое значение из метода или функции, которые могут препятствовать обнаружению неожиданных состояний и условий.

CWE-253: неправильная проверка возвращаемого значения функции - программа неправильно проверяет возвращаемое значение из функции, что не позволяет программному обеспечению обнаруживать ошибки или исключительные условия.

CWE-256: Незащищенное хранение учетных данных. Хранение пароля в виде открытого текста может привести к компрометации системы.

```
char * username = "User";
char * domain = "Domain";
/* POTENTIAL FLAW: Attempt to login user with password from the source */
if (LogonUserA(
    username,
    domain,
    data,
    LOGON32_LOGON_NETWORK,
    LOGON32_PROVIDER_DEFAULT,
    &pHandle) != 0)
```

CWE-259: Использование жестко запрограммированного пароля. Программное обеспечение содержит жестко запрограммированный пароль, который используется для собственной входящей аутентификации или для исходящей связи с внешними компонентами.

CWE-272: Нарушение прав - повышенный уровень привилегий, необходимый для выполнения таких операций, как chroot (), должен быть сброшен сразу после выполнения операции.

CWE-273: Неправильная проверка привилегий. Программное обеспечение пытается отбросить привилегии, но не проверяет или неправильно проверяет, удалось ли предоставить.

```
/* FLAW: Failed to check return status of ImpersonateNamedPipeClient
 * -- However, since we're not even DOING anything with the pipe
 * it's debatable whether this is really a bug
 */
ImpersonateNamedPipeClient(hPipe);
printLine("Impersonated");
if (!RevertToSelf())
{
    exit(1);
}
```

CWE-284: Неправильный контроль доступа - Программное обеспечение не ограничивает или неправильно ограничивает доступ к ресурсу от неавторизованного участника.

CWE-319: Передача конфиденциальной информации в открытом тексте - программное обеспечение передает конфиденциальные или критически важные данные в виде открытого текста по каналу связи, который может быть прослушан неуполномоченными лицами.

CWE-321: Использование жестко закодированного криптографического ключа. Использование жестко закодированного криптографического ключа значительно увеличивает вероятность восстановления зашифрованных данных.

CWE-325: Отсутствует необходимый криптографический шаг - программное обеспечение не реализует требуемый шаг в криптографическом алгоритме, что приводит к более слабому шифрованию, чем объявлено этим алгоритмом.

```
/* Copy plaintext into payload buffer */
memcpy(payload, PAYLOAD, payloadLen);
/* Aquire a Context */
if(!CryptAcquireContext(&hCryptProv, NULL, MS_ENH_RSA_AES_PROV, PROV_RSA_AES, 0))
{
    break;
}
/* FLAW: Missing required step (CryptCreateHash) causes the payload to remain in plaintext form */
/* Hash the input string */
if(!CryptHashData(hHash, (BYTE*)hashData, strlen(hashData), 0))
{
    break;
}
```

CWE-327: Использование сломанного или рискованного криптографического алгоритма - Использование сломанного или рискованного криптографического алгоритма - это ненужный риск, который может привести к раскрытию конфиденциальной информации.

CWE-328: Обратимое одностороннее хеширование - продукт использует алгоритм хеширования, который выдает значение хеш-функции, которое можно использовать для определения исходного ввода или для поиска ввода, которое может произвести такое же хеширование, более эффективно, чем методы перебора.

CWE-338: Использование криптографически слабого генератора псевдослучайных чисел (PRNG) - Продукт использует генератор псевдослучайных чисел (PRNG) в контексте безопасности, но алгоритм PRNG не является криптографически стойким.

CWE-364: Состояние состязания обработчика сигнала - Программное обеспечение использует обработчик сигнала, который вводит условие состязания.

CWE-366: Состояние гонки в потоке - если два потока выполнения используют ресурс одновременно, существует вероятность того, что ресурсы могут быть использованы как недопустимые, что, в свою очередь, делает состояние выполнения неопределенным.

CWE-367: Состояние гонки во время проверки (TOCTOU). Состояние гонки - программное обеспечение проверяет состояние ресурса перед использованием этого ресурса, но состояние ресурса может меняться между проверкой и использованием таким способом, который делает недействительным результаты проверки. Это может привести к тому, что программное обеспечение выполнит недопустимые действия, когда ресурс находится в непредвиденном состоянии.

CWE-369: Divide By Zero - продукт делит значение на ноль.

CWE-377: Небезопасный временный файл. Создание и использование небезопасных временных файлов может сделать данные приложения и системы уязвимыми для атаки.

CWE-390: обнаружение состояния ошибки без действий - программное обеспечение обнаруживает конкретную ошибку, но не предпринимает никаких действий для устранения ошибки.

CWE-396: Декларация улова для общего исключения - отлов слишком широких исключений способствует сложному коду обработки ошибок, который с большей вероятностью содержит уязвимости безопасности.

CWE-397: Декларация о создании исключений общего характера - Создание слишком широких исключений способствует сложному коду обработки ошибок, который с большей вероятностью содержит уязвимости безопасности.

```
/* maintenance note: "throw (...)" is accepted by Visual Studio, but not gcc,  
 * which is why this test case is windows specific */  
void bad() throw (...) /* FLAW: declaring that you throw some type of exception, but without specifics */  
{  
    throw range_error("Test");  
}
```

CWE-400: Неконтролируемое потребление ресурсов. Программное обеспечение неправильно контролирует распределение и обслуживание ограниченного ресурса, что позволяет субъекту влиять на количество потребляемых ресурсов, что в конечном итоге приводит к исчерпанию доступных ресурсов.

CWE-401: Отсутствие освобождения памяти после истечения срока - программное обеспечение недостаточно отслеживает и освобождает выделенную память после ее использования, что медленно потребляет оставшуюся память.

CWE-404: Неправильное отключение или освобождение ресурса - программа не освобождает или неправильно освобождает ресурс, прежде чем он станет доступным для повторного использования.

CWE-415: Double Free - продукт дважды вызывает free () по одному и тому же адресу памяти, что может привести к изменению неожиданных областей памяти.

CWE-416: Использовать после освобождения - ссылка на память после ее освобождения может вызвать сбой программы, использование неожиданных значений или выполнение кода.

CWE-426: Ненадежный путь поиска - приложение выполняет поиск критических ресурсов, используя предоставленный извне путь поиска, который может указывать на ресурсы, которые не находятся под непосредственным контролем приложения.

CWE-427: Элемент неконтролируемого пути поиска - Продукт использует фиксированный или контролируемый путь поиска для поиска ресурсов, но одно или несколько мест в этом пути могут находиться под контролем непреднамеренных участников.

CWE-440: Ожидаемое нарушение поведения - функция, API или процедура, используемые продуктом, ведут себя не так, как ожидает продукт.

CWE-457: Использование неинициализированной переменной - в коде используется переменная, которая не была инициализирована, что приводит к непредсказуемым или непреднамеренным результатам.

CWE-459: Неполная очистка - программное обеспечение неправильно «очищает» и удаляет временные или вспомогательные ресурсы после их использования.

```
pFile = FDOPEN(fileDesc, "w");
if (pFile != NULL)
{
    fprintf(pFile, "Temporary file");
    fclose(pFile);
    /* FLAW: We don't unlink */
}
```

CWE-464: добавление Sentinel структуры данных. Случайное добавление Sentinel структуры данных может вызвать серьезные проблемы в логике программирования.

CWE-467: Использование sizeof () для типа указателя - Код вызывает sizeof () для типа неверного указателя, который всегда возвращает значение wordsize / 8. Это может привести к неожиданному результату, если программист намеревался определить, сколько памяти было выделено.

CWE-468: неправильное масштабирование указателя - в C и C ++ часто можно случайно сослаться на неправильную память из-за семантики, когда математические операции неявно масштабируются.

CWE-469: Использование вычитания указателя для определения размера - приложение вычитает один указатель из другого, чтобы определить размер, но этот расчет может быть неправильным, если указатели не существуют в одном и том же фрагменте памяти.

CWE-475: Неопределенное поведение для ввода в API. Поведение этой функции не определено, если для ее управляющего параметра не установлено конкретное значение.

```
char dataBuffer[100] = "";
char * data = dataBuffer;
strcpy(data, "abcdefghijklmnopqrstuvwxyz");
/* FLAW: Copy overlapping memory regions using memcpy() for which the result is undefined */
memcpy(data + 6, data + 4, 10*sizeof(char));
printLine(data);
```

CWE-476: разыменованное указателя NULL - разыменованное указателя NULL происходит, когда приложение разыменовывает указатель, который, как он ожидает, является допустимым, но имеет значение NULL, обычно вызывая сбой или выход.

CWE-478: отсутствует регистр по умолчанию в операторе переключения - в коде нет регистра по умолчанию в операторе переключения, что может привести к сложным логическим ошибкам и вытекающим из этого слабостям.

CWE-479: Использование обработчиком сигнала не-реентерабельной функции - Программа определяет обработчик сигнала, который вызывает не-реентерабельную функцию.

CWE-480: Использование неверного оператора - Программист случайно использует неправильный оператор, который изменяет логику приложения с точки зрения безопасности.

CWE-481: присвоение вместо сравнения - код использует оператор для присвоения, когда целью было выполнить сравнение.

CWE-482: Сравнение вместо присвоения - код использует оператор для сравнения, когда предполагалось выполнить присвоение.



CWE-483: Неправильное разделение блоков - код явно не ограничивает блок, который должен содержать 2 или более операторов, создавая логическую ошибку.

CWE-484: пропущенный оператор прерывания в коммутаторе - Программа пропускает оператор прерывания в коммутаторе или аналогичной конструкции, вызывая выполнение кода, связанного с несколькими условиями. Это может вызвать проблемы, когда программист только намеревался выполнить код, связанный с одним условием.

CWE-500: Публичное статическое поле не помечено как финальное - объект содержит общедоступное статическое поле, которое не помечено как финальное, что может привести к его неожиданным изменениям.

CWE-506: Встроенный вредоносный код - приложение содержит код, который по своей природе является вредоносным.

```
/* FLAW: Send the contents of a file over the network */
if (send(connectSocket, contents, strlen(contents), 0) != strlen(contents))
{
    break;
}
```

CWE-510: Потайной ход - это скрытый кусок кода, который реагирует на специальный ввод, предоставляя пользователю доступ к ресурсам без прохождения через обычный механизм обеспечения безопасности.

CWE-511: Логика / бомба замедленного действия - программное обеспечение содержит код, предназначенный для нарушения законной работы программного обеспечения (или его среды) по истечении определенного времени или при соблюдении определенного логического условия.

CWE-526: Доступ к информации через переменные среды. Переменные среды могут содержать конфиденциальную информацию об удаленном сервере.

```
void CWE526_Info_Exposure_Environment_Variables__basic_01_bad()
{
    /* FLAW: environment variable exposed */
    printLine(getenv("PATH"));
}
```

CWE-535: Сообщение об ошибке при передаче информации через оболочку. Сообщение об ошибке командной оболочки указывает на наличие необработанного исключения в коде веб-приложения. Во многих случаях злоумышленник может использовать условия, которые вызывают эти ошибки, чтобы получить несанкционированный доступ к системе.

CWE-546: Подозрительный комментарий - код содержит комментарии, которые указывают на наличие ошибок, неполной функциональности или слабых мест.

CWE-561: Мертвый код - программное обеспечение содержит мертвый код, который никогда не может быть выполнен.

CWE-562: Возврат адреса переменной стека - функция возвращает адрес переменной стека, что приведет к непреднамеренному поведению программы, обычно в форме сбоя.

CWE-563: Присвоение переменной без использования - значение переменной присваивается, но никогда не используется, что делает его мертвым хранилищем.

CWE-570: Выражение всегда ложно - программное обеспечение содержит выражение, которое всегда будет иметь значение false.

CWE-571: Выражение всегда истинно - программное обеспечение содержит выражение, которое всегда будет иметь значение true.

CWE-587: Присвоение фиксированного адреса указателю - программное обеспечение устанавливает указатель на определенный адрес, отличный от NULL или 0.

CWE-588: Попытка доступа к дочернему элементу неструктурного указателя - приведение неструктурного типа к структурному типу и доступ к полю могут привести к ошибкам доступа к памяти или повреждению данных.

CWE-590: Свободной памяти нет в куче - приложение вызывает функцию free () для указателя на память, которая не была выделена с использованием связанных функций выделения кучи, таких как malloc (), calloc () или realloc ().

CWE-591: Хранение конфиденциальных данных в неправильно заблокированной памяти - приложение хранит конфиденциальные данные в памяти, которая не заблокирована или неправильно заблокирована, что может привести к записи памяти для замены файлов на диске диспетчером виртуальной памяти. Это может сделать данные более доступными для внешних участников.

```
/* FLAW: Do not lock the memory */
/* INCIDENTAL FLAW: CWE-259 Hardcoded Password */
strcpy(password, "Password1234!");
{
    HANDLE pHandle;
    char * username = "User";
    char * domain = "Domain";
    /* Use the password in LogonUser() to establish that it is "sensitive" */
    if (LogonUserA(
        username,
        domain,
        password,
        LOGON32_LOGON_NETWORK,
        LOGON32_PROVIDER_DEFAULT,
        &pHandle) != 0)
```

CWE-605: множественные привязки к одному и тому же порту. Если разрешено связывание нескольких сокетов с одним и тем же портом, другие службы на этом порту могут быть украдены или подделаны.

CWE-606: Неконтролируемый вход для условия цикла - продукт неправильно проверяет входы, которые используются для условий цикла, что может привести к отказу в обслуживании из-за чрезмерно большого цикла.

CWE-615: Раскрытие информации посредством комментариев. Хотя добавление общих комментариев очень полезно, некоторые программисты склонны оставлять важные данные, такие как: имена файлов, связанные с веб-приложением, старые ссылки или ссылки, которые не предназначались для просмотра пользователями, старые фрагменты кода и т. д.

CWE-617: Reachable Assertion - продукт содержит assert () или аналогичное утверждение, которое может быть запущено злоумышленником, что приводит к завершению работы приложения или другому поведению, более серьезному, чем необходимо.

```
/* POTENTIAL FLAW: this assertion could trigger if n <= ASSERT_VALUE */
assert(data > ASSERT_VALUE);
```

CWE-620: Непроверенная смена пароля - при установке нового пароля для пользователя продукт не требует знания исходного пароля или использования другой формы аутентификации.

CWE-665: Неправильная инициализация - программное обеспечение не инициализирует или неправильно инициализирует ресурс, что может привести к непредвиденному состоянию ресурса при доступе или использовании.

CWE-666: Операция с ресурсом в неправильной фазе срока службы. Программное обеспечение выполняет операцию с ресурсом на неправильной фазе жизненного цикла ресурса, что может привести к неожиданному поведению.

CWE-667: Неправильная блокировка - программное обеспечение неправильно устанавливает блокировку ресурса или неправильно снимает блокировку ресурса, что приводит к неожиданным изменениям состояния ресурса и поведением.

CWE-672: Операция с ресурсом после истечения срока его действия или освобождения. Программное обеспечение использует ресурс, обращается к нему или иным образом работает с ресурсом после истечения срока его действия, освобождения или отзыва.

CWE-674: Неконтролируемая рекурсия. Продукт неправильно контролирует количество выполняемой рекурсии, которая потребляет чрезмерные ресурсы, такие как выделенная память или стек программ.

```
static void helperBad()
{
    /* FLAW: this function causes infinite recursion */
```

```

    helperBad(); /* maintenance note: this may generate a warning, this is on purpose */
}

void CWE674_Uncontrolled_Recursion__infinite_recursive_call_01_bad()
{
    helperBad();
}

```

CWE-675: Дублирующиеся операции над ресурсом - продукт выполняет одну и ту же операцию над ресурсом два или более раз, когда операцию следует применять только один раз.

CWE-676: Использование потенциально опасной функции - программа вызывает потенциально опасную функцию, которая может привести к уязвимости, если она используется неправильно, но функцию также можно использовать безопасно.

CWE-680: Целочисленное переполнение для переполнения буфера - продукт выполняет вычисление, чтобы определить объем выделяемой памяти, но может произойти целочисленное переполнение, в результате которого выделяется меньше памяти, чем ожидалось, что приводит к переполнению буфера.

CWE-681: Неправильное преобразование между числовыми типами - при преобразовании из одного типа данных в другой, например, long в целое число, данные могут быть опущены или преобразованы таким образом, что получаются неожиданные значения. Если полученные значения используются в чувствительном контексте, то может возникнуть опасное поведение.

CWE-685: Вызов функции с неверным количеством аргументов. Программное обеспечение вызывает функцию, процедуру или процедуру, но вызывающая сторона указывает слишком много аргументов или слишком мало аргументов, что может привести к неопределенному поведению и вытекающим из этого слабостям.

CWE-688: Вызов функции с неверной переменной или ссылкой в качестве аргумента - программное обеспечение вызывает функцию, процедуру или процедуру, но вызывающая сторона указывает неверную переменную или ссылку в качестве одного из аргументов, что может привести к неопределенному поведению и вытекающим из этого слабостям.

CWE-690: Непроверенное возвращаемое значение для разыменования указателя NULL - продукт не проверяет наличие ошибки после вызова функции, которая может возвращать с указателем NULL в случае сбоя функции, что приводит к разыменованию результирующего указателя NULL.

```

void CWE690_NULL_Deref_From_Return__char_calloc_01_bad()
{
    char * data;
    data = NULL; /* Initialize data */
    /* POTENTIAL FLAW: Allocate memory without checking if the memory allocation function failed */
    data = (char *)calloc(20, sizeof(char));
    /* FLAW: Initialize memory buffer without checking to see if the memory allocation function failed */
    strcpy(data, "Initialize");
    printLine(data);
    free(data);
}

```

CWE-758: Опора на неопределенное или определяемое реализацией поведение - программное обеспечение использует функцию API, структуру данных или другую сущность таким образом, что полагается на свойства, которые не всегда гарантированно сохраняются для этой сущности.

CWE-761: Указатель свободен, а не в начале буфера - приложение вызывает функцию free () для указателя на ресурс памяти, который был выделен в куче, но указатель не находится в начале буфера.

CWE-762: Несоответствующие процедуры управления памятью. Приложение пытается вернуть ресурс памяти в систему, но вызывает функцию освобождения, которая несовместима с функцией, которая первоначально использовалась для выделения этого ресурса.

CWE-773: Отсутствует ссылка или дескриптор активного файла - программное обеспечение неправильно поддерживает ссылки на дескриптор файла, что препятствует восстановлению дескриптора файла.

CWE-775: Отсутствует дескриптор файла после истечения срока действия - программное обеспечение не освобождает дескриптор файла после истечения срока его использования, то есть после того, как дескриптор файла больше не требуется.

CWE-780: использование алгоритма RSA без OAEP. Программное обеспечение использует алгоритм RSA, но не включает в себя оптимальное асимметричное заполнение шифрования (OAEP), что может ослабить шифрование.

CWE-785: Использование функции манипулирования путями без буфера максимального размера. Программное обеспечение вызывает функцию для нормализации путей или имен файлов, но предоставляет выходной буфер, который меньше максимально возможного размера, например PATH\_MAX.

CWE-789: Неконтролируемое выделение памяти - продукт распределяет память на основе ненадежного значения размера, но не проверяет или неправильно проверяет размер, позволяя выделять произвольные объемы памяти.

CWE-832: Разблокировка ресурса, который не заблокирован - программа пытается разблокировать ресурс, который не заблокирован.

CWE-835: Цикл с недостижимым условием выхода («бесконечный цикл») - программа содержит итерацию или цикл с условием выхода, которое не может быть достигнуто, то есть бесконечный цикл.

CWE-843: Доступ к ресурсу с использованием несовместимого типа («путаница типов») - программа выделяет или инициализирует ресурс, такой как указатель, объект или переменную, используя один тип, но позднее получает доступ к этому ресурсу, используя тип, который несовместим с оригинальным типом.

```
void CWE843_Type_Confusion__char_01_bad()
{
    void * data;
    /* Initialize data */
    data = NULL;
    {
        /* FLAW: Point data to a char */
        char charBuffer = 'a';
        data = &charBuffer;
    }
    /* POTENTIAL FLAW: Attempt to access data as an int */
    printIntLine(*(int*)data);
}
```

## Flawfinder

Flawfinder работает с использованием встроенной базы данных функций C / C++ с хорошо известными проблемами, такими как риски переполнения буфера (например, `strcpy()`, `strcat()`, `gets()`, `sprintf()` и семейство `scanf()`), проблемы форматной строки (`[v] [f] printf()`, `[v] snprintf()` и `syslog()`), условия гонки (такие как `access()`, `chown()`, `chgrp()`, `chmod()`, `tmpfile()`, `tmpnam()`, `tempnam()` и `mktemp()`), потенциальные опасности метасимволов оболочки (большая часть семейства `exec()`, `system()`, `popen()`) и плохое получение случайных чисел (например, `random()`)).

Затем Flawfinder берет текст исходного кода и сопоставляет текст исходного кода с этими именами, игнорируя при этом текст внутри комментариев и строк (за исключением директив `flawfinder`). Flawfinder также использует `gettext` (общая библиотека для интернационализированных программ) и будет обрабатывать значения, передаваемые через `gettext`, как если бы они были константами; это уменьшает количество ложных попаданий в интернационализированных программах.

Flawfinder создает список «совпадений» (потенциальных недостатков безопасности), отсортированных по степени риска; по умолчанию самые рискованные попадания показываются первыми. Этот уровень риска зависит не только от функции, но и от значений параметров функции. Например, постоянные строки часто менее рискованны, чем полностью переменные строки во многих контекстах. В некоторых случаях дефектоскоп может определить, что конструкция вообще не опасна, что снижает количество ложных срабатываний.

Flawfinder дает лучшую информацию - и лучшую расстановку приоритетов - чем просто «grep» для исходного кода. В конце концов, он знает, что должен игнорировать комментарии и внутреннюю часть строк, а также проверяет параметры для оценки уровней риска. Тем не менее, `flawfinder` - это принципиально наивная программа; он даже не знает о типах данных параметров функций и, конечно, не выполняет анализ потока управления или анализа потока данных (см. ссылки ниже на другие инструменты, такие как `SPLINT`, которые выполняют более глубокий анализ). Кроме того, поскольку он прост, его не смущают определения макросов и другие странности, с которыми сталкиваются более сложные инструменты.

Flawfinder не понимает семантику кода - он в основном выполняет простое сопоставление текстовых шаблонов (игнорируя комментарии и строки). Опять же, он не выполняет поток данных или анализ потока управления вообще. Тем не менее, `flawfinder` может быть очень полезным средством поиска и устранения уязвимостей в безопасности.

```
perform an over-read (it could cause a crash if unprotected) (CWE-126).
CWE126_Buffer_Overread/s03/CWE126_Buffer_Overread__wchar_t_declare_memmove_81_g
oodG2B.cpp:32: [1] (buffer) wcslen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).
CWE126_Buffer_Overread/s03/CWE126_Buffer_Overread__wchar_t_declare_memmove_82_b
ad.cpp:32: [1] (buffer) wcslen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).
CWE126_Buffer_Overread/s03/CWE126_Buffer_Overread__wchar_t_declare_memmove_82_g
oodG2B.cpp:32: [1] (buffer) wcslen:
Does not handle strings that are not \0-terminated; if given one it may
perform an over-read (it could cause a crash if unprotected) (CWE-126).

ANALYSIS SUMMARY:

Hits = 9515
Lines analyzed = 278043 in approximately 17.70 seconds (15705 lines/second)
Physical Source Lines of Code (SLOC) = 229517
Hits@level = [0] 0 [1] 2550 [2] 5508 [3] 1457 [4] 0 [5] 0
Hits@level+ = [0+] 9515 [1+] 9515 [2+] 6965 [3+] 1457 [4+] 0 [5+] 0
Hits/KSLOC@level+ = [0+] 41.4566 [1+] 41.4566 [2+] 30.3463 [3+] 6.34811 [4+]
0 [5+] 0
Minimum risk level = 1
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
See 'Secure Programming for Linux and Unix HOWTO'
(http://www.dwheeler.com/secure-programs) for more information.
dkulikov@ubuntu:~/Pictures/C/testcases$
```

Рисунок 1 – работа flawfinder

## Cppcheck

Это анализатор исходного кода, который обнаруживает различные виды ошибок и уязвимостей.

Доступны как интерфейс командной строки, так и графический интерфейс пользователя.

Cppcheck уделяет большое внимание обнаружению неопределенного поведения.

- Мертвые указатели
- Деление на ноль
- Целочисленные переполнения
- Неверные битовые операнды
- Неверные преобразования
- Неправильное использование STL
- Управление памятью
- Разыменование нулевого указателя
- Выход за границы
- Неинициализированные переменные
- Запись постоянных данных

Наиболее распространенными типами уязвимостей в 2017 году (количество CVE) были:

Категория	Количество	Обнаруженные Cppcheck
Ошибки буфера	2530	Несколько
Неправильный контроль доступа	1366	Непреднамеренные потайные ходы
Утечка информации	1426	Непреднамеренные потайные ходы
Разрешения, привилегии и контроль доступа	1196	Непреднамеренные потайные ходы
Проверка ввода	968	Нет

CVE, которые были найдены с помощью Cppcheck:

CVE-2017-1000249: файл: переполнение буфера в стеке. Это было найдено Томасом Ярошем с помощью Cppcheck. Причина - ошибка в условии.

CVE-2013-6462: 23-летняя уязвимость переполнения стека в X.org, обнаруженная с помощью Cppcheck.

CVE-2012-1147: readfilemap.c в expat 2.1.0 позволяет злоумышленникам, вызывать отказ в обслуживании (отказ в использовании дескриптора файла) через большое количество специально созданных XML-файлов.

Cppcheck сначала полностью обрабатывает исходные файлы препроцессором. Предобработанный код токенизируется, и проверки осуществляются над каждым токеном в цикле. По токенам строятся синтаксические деревья. Также используется база данных о типах, функциях и так далее. Программа использует ее когда встречается такой токен в исходном коде. Например, проверка на деление на ноль выглядит следующим образом:

```
// We have a token, is it a division operator?
if (token>str() == "/") {
    // Get RHS operand of division operator
    const Token *rhs = token->astOperand2();
    // Get "0" value of RHS operand if it exists
    const ValueFlow::Value *val0 = rhs->getValue(0);
    // Is there a "0" value for RHS?
    if (val0 != NULL) {
        .. yes there is a "0" value of RHS ..
    }
}
```

```
[CWE122_Heap_Based_Buffer_Overflow/s11/CWE122_Heap_Based_Buffer_Overflow__sizeo
f_struct_74a.cpp:38]: (warning) Size of pointer 'data' used instead of size of
its data.
[CWE122_Heap_Based_Buffer_Overflow/s11/CWE122_Heap_Based_Buffer_Overflow__sizeo
f_struct_81a.cpp:32]: (warning) Size of pointer 'data' used instead of size of
its data.
[CWE122_Heap_Based_Buffer_Overflow/s11/CWE122_Heap_Based_Buffer_Overflow__sizeo
f_struct_82a.cpp:32]: (warning) Size of pointer 'data' used instead of size of
its data.
[CWE122_Heap_Based_Buffer_Overflow/s11/CWE122_Heap_Based_Buffer_Overflow__sizeo
f_struct_83_bad.cpp:28]: (warning) Size of pointer 'data' used instead of size
of its data.
[CWE122_Heap_Based_Buffer_Overflow/s11/CWE122_Heap_Based_Buffer_Overflow__sizeo
f_struct_84_bad.cpp:28]: (warning) Size of pointer 'data' used instead of size
of its data.
```

Рисунок 2 – работа cppcheck

## RATS

Анализатор исходного кода RATS

RATS был создан Fortify и предназначен для поиска ошибок в коде, написанном на C / C ++, Perl, Ruby, PHP и Python, и распространяется бесплатно. По работе он идентичен Flawfinder.

Rats 2.3 имеет следующее количество паттернов в своей базе данных:

Entries in perl database: 33

Entries in ruby database: 46

Entries in python database: 62

Entries in c database: 334

Entries in php database: 55

```
CWE114_Process_Control/CWE114_Process_Control_w32_char_relativePath_83_bad.cpp:38: High: LoadLibraryA
CWE114_Process_Control/CWE114_Process_Control_w32_char_relativePath_83_goodG2B.cpp:38: High: LoadLibraryA
CWE114_Process_Control/CWE114_Process_Control_w32_char_relativePath_84_bad.cpp:38: High: LoadLibraryA
CWE114_Process_Control/CWE114_Process_Control_w32_char_relativePath_84_goodG2B.cpp:38: High: LoadLibraryA
LoadLibrary will search several places for a library if
no path is specified, allowing trojan DLL's to be inserted elsewhere even
if the intended DLL is correctly protected from overwriting. Make sure to specify the full path.
```

Рисунок 3 – работа RATS

Таблица 1 – Результаты тестирования анализаторов

CWE	Flawfinder	Cppcheck	Rats
CWE15	-	-	-
CWE23	-	-	-
CWE36	-	-	-
CWE78	+	-	+
CWE90	-	-	-
CWE114	+	-	+
CWE121	+	+	+
CWE122	+	+	+



CWE123	-	-	-
CWE124	-	+	-
CWE126	-	+	-
CWE127	-	+	-
CWE134	+	-	+
CWE176	-	-	-
CWE188	-	-	-
CWE190	-	-	-
CWE191	-	-	-
CWE194	-	+	-
CWE195	-	-	-
CWE196	-	-	-
CWE197	-	-	-
CWE222	-	-	-
CWE223	-	-	-
CWE226	-	-	-
CWE242	+	+	+
CWE244	-	-	-
CWE252	-	-	-
CWE253	-	-	-
CWE256	-	-	-
CWE259	-	-	-
CWE272	-	-	-
CWE273	-	-	-
CWE284	-	-	-
CWE319	-	-	-
CWE321	-	-	-
CWE325	-	-	-
CWE327	-	-	-
CWE328	-	-	-
CWE338	+	-	+
CWE364	-	-	-
CWE366	-	-	-
CWE367	-	-	-
CWE369	-	+	-
CWE377	-	-	-
CWE390	-	+	-
CWE396	-	-	-
CWE397	-	-	-
CWE400	-	-	-
CWE401	+	+	+
CWE404	-	-	-
CWE415	-	+	-
CWE416	-	-	-
CWE426	-	-	-
CWE427	-	-	-
CWE440	-	-	-
CWE457	-	+	-
CWE459	-	-	-
CWE464	-	-	-

CWE467	-	+	-
CWE468	-	-	-
CWE469	-	-	-
CWE475	-	-	-
CWE476	-	+	-
CWE478	-	-	-
CWE479	+	-	+
CWE480	-	-	-
CWE481	-	-	-
CWE482	-	-	-
CWE483	-	-	-
CWE484	-	-	-
CWE500	-	-	-
CWE506	-	-	-
CWE510	-	-	-
CWE511	-	-	-
CWE526	+	-	+
CWE535	-	-	-
CWE546	-	-	-
CWE561	-	-	-
CWE562	-	+	-
CWE563	-	+	-
CWE570	-	-	-
CWE587	-	-	-
CWE588	-	-	-
CWE590	-	+	-
CWE591	-	-	-
CWE605	-	-	-
CWE615	-	-	-
CWE617	-	-	-
CWE620	-	-	-
CWE665	-	-	-
CWE666	-	-	-
CWE667	-	-	-
CWE672	-	-	-
CWE674	-	-	-
CWE675	-	+	-
CWE676	-	-	-
CWE680	-	-	-
CWE681	-	-	-
CWE685	-	+	-
CWE688	-	+	-
CWE690	-	+	-
CWE758	-	+	-
CWE761	-	-	-
CWE762	-	+	-
CWE773	-	-	-
CWE775	-	+	-
CWE780	-	-	-
CWE785	+	-	+

CWE789	-	-	-
CWE832	-	-	-
CWE835	-	-	-
CWE843	-	+	-

## Clang

Clang — это транслятор для C-подобных языков, созданный специально для работы на базе LLVM. Комбинация Clang и LLVM представляет собой полноценный компилятор и предоставляет набор инструментов, позволяющих полностью заменить GCC. Благодаря архитектуре, основанной на библиотеках, Clang (как и LLVM) легко встраивается в другие приложения.

В отличие от GCC, Clang изначально спроектирован для максимального сохранения информации в ходе процесса компиляции, в том числе сохранения «внешнего вида» исходного кода. Эта особенность позволяет Clang создавать развернутые контекстно-ориентированные сообщения об ошибках, понятные как для программистов, так и для сред разработки. Модульный дизайн компилятора позволяет использовать его в составе среды разработки для индексирования кода, подсветки синтаксиса и переработки кода.

В большинстве случаев, Clang запускает препроцессор (который разворачивает все макросы) и парсит исходник, превращая его в абстрактное синтаксическое дерево (AST). С AST работать гораздо проще, чем с исходным кодом, но всегда можно получить ссылки на исходник. Фактически, каждая структура в Clang-е, используемая для представления кода (AST, CFG и т.п.), всегда имеет ссылку на оригинальный исходник, полезный для целей анализа, рефакторинга и т.п.



```

'myUnion.unionFirst'
char * data = myUnion.unionSecond;
1 warning generated.
testcases/CWE590_Free_Memory_Not_on_Heap/s01/CWE590_Free_Memory_Not_on_Heap__de
lete_array_char_static_41.cpp:30:5: warning:
  Argument to 'delete[]' is the address of the static
  variable 'dataBuffer', which is not memory allocated
  by 'new[]'
delete [] data;

```

Рисунок 4 – работа clang

Clang достаточно популярный инструмент, более сложный в использовании, но способный обнаруживать ошибки, другие анализаторы которые обнаружить не могут, например CWE761:

```

4 warnings generated.
testcases/CWE761_Free_Pointer_Not_at_Start_of_Buffer/CWE761_Free_Pointer_Not_at
_Start_of_Buffer__char_connect_socket_16.c:131:9: warning:
    Argument to free() is offset by 1 byte from the start of memory allocated
    by malloc()
    free(data);
    ^~~~~~

testcases/CWE761_Free_Pointer_Not_at_Start_of_Buffer/CWE761_Free_Pointer_Not_at
_Start_of_Buffer__char_connect_socket_16.c:131:9: warning:
    Argument to free() is offset by 2 bytes from the start of memory
    allocated by malloc()
    free(data);
    ^~~~~~

testcases/CWE761_Free_Pointer_Not_at_Start_of_Buffer/CWE761_Free_Pointer_Not_at
_Start_of_Buffer__char_connect_socket_16.c:131:9: warning:
    Argument to free() is offset by 3 bytes from the start of memory
    allocated by malloc()
    free(data);
    ^~~~~~

```

Рисунок 5 – обнаружение CWE761

## Bitcoin

В качестве open source проекта для тестирования анализаторов был выбран Bitcoin.

Результат работы RATS:

```

src/leveldb/util/testharness.cc:68: High: getenv
src/qt/guiutil.cpp:630: High: getenv
src/qt/guiutil.cpp:632: High: getenv
src/test/util_tests.cpp:691: High: getenv
src/test/util_tests.cpp:826: High: getenv
src/util/system.cpp:697: High: getenv
Environment variables are highly untrustable input. They may be of any length, and contain any data. Do
not make any assumptions regarding content or length. If at all possible avoid using them, and if it is
necessary, sanitize them and truncate them to a reasonable length.

src/leveldb/port/port_win.cc:58: High: EnterCriticalSection
This function can throw exceptions in low memory
conditions. Use InitialCriticalSectionAndSpinCount instead.

src/leveldb/util/env_win.cc:748: High: GetTempPathW
GetTempPath() may return the current directory or the
windows directory. Be careful what you place in these locations. Important
files may be overwritten, and trojan DLL's may be dropped in these
locations. Never use a user-input filename when writing to a location given
by GetTempPath().

src/secp256k1/src/bench_ecmult.c:132: High: sprintf
Check to be sure that the non-constant format string passed as argument 2 to
this function call does not come from an untrusted source that could have added
formatting characters that the code is not prepared to handle.

src/secp256k1/src/bench_ecmult.c:132: High: sprintf
Check to be sure that the format string passed as argument 2 to this function
call does not come from an untrusted source that could have added formatting
characters that the code is not prepared to handle. Additionally, the format
string could contain '%s' without precision that could result in a buffer
overflow.

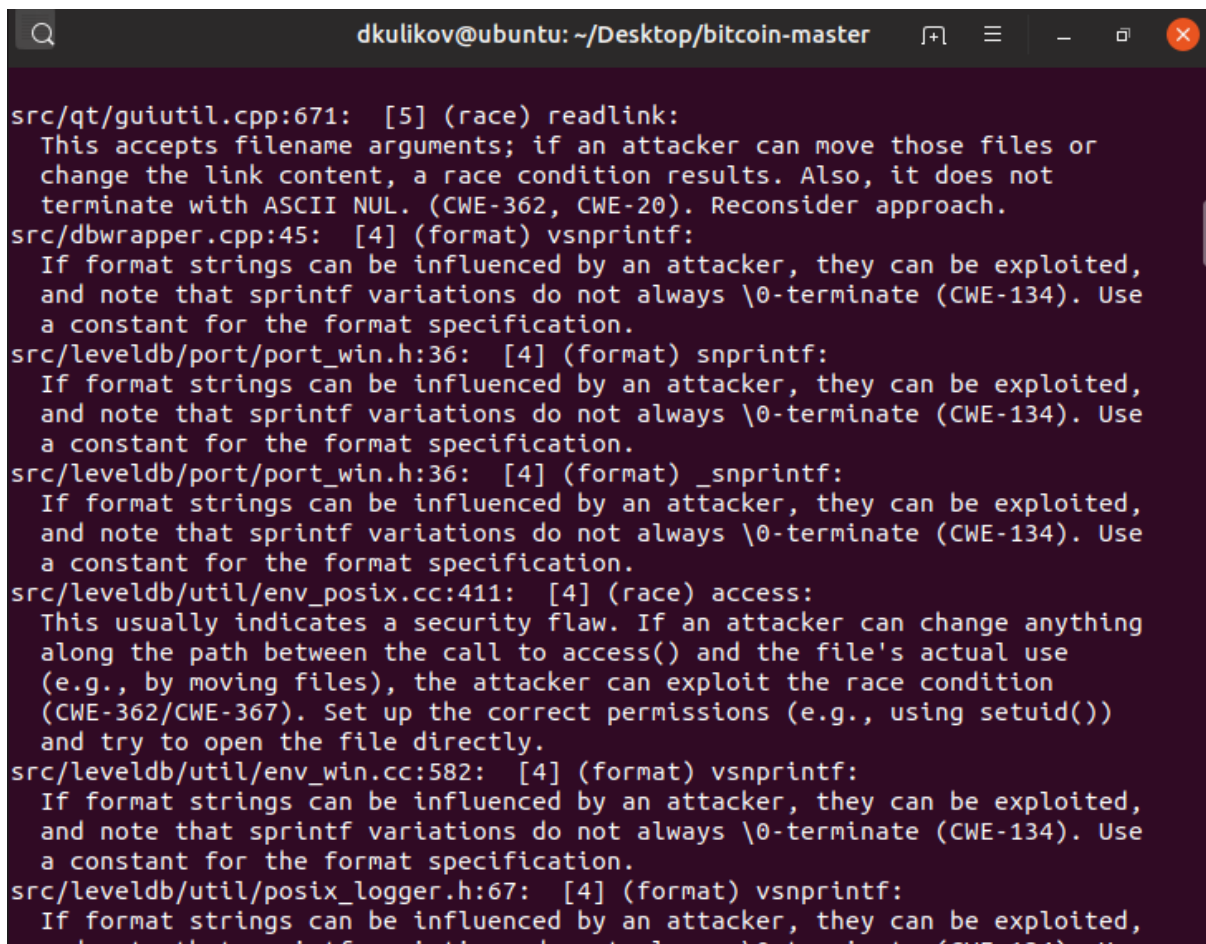
src/util/system.cpp:1125: High: system
Argument 1 to this function call should be checked to ensure that it does not
come from an untrusted source without first verifying that it contains nothing
dangerous.

src/util/system.cpp:1127: High: _wsystem
Many program execution commands under Windows will search
the path for a program if you do not explicitly specify a full path to the
file. This can allow trojans to be executed instead. Also, be sure to
specify a file extension, since otherwise multiple extensions will be tried
by the operating system, providing another opportunity for trojans.

src/util/system.cpp:1127: High: _wsystem
Argument 1 to this function call should be checked to ensure that it does not
come from an untrusted source without first verifying that it contains nothing
dangerous.

```

Рисунок 6 – результат работы RATS



```
src/qt/guiutil.cpp:671: [5] (race) readlink:
  This accepts filename arguments; if an attacker can move those files or
  change the link content, a race condition results. Also, it does not
  terminate with ASCII NUL. (CWE-362, CWE-20). Reconsider approach.
src/dbwrapper.cpp:45: [4] (format) vsnprintf:
  If format strings can be influenced by an attacker, they can be exploited,
  and note that sprintf variations do not always \0-terminate (CWE-134). Use
  a constant for the format specification.
src/leveldb/port/port_win.h:36: [4] (format) snprintf:
  If format strings can be influenced by an attacker, they can be exploited,
  and note that sprintf variations do not always \0-terminate (CWE-134). Use
  a constant for the format specification.
src/leveldb/port/port_win.h:36: [4] (format) _snprintf:
  If format strings can be influenced by an attacker, they can be exploited,
  and note that sprintf variations do not always \0-terminate (CWE-134). Use
  a constant for the format specification.
src/leveldb/util/env_posix.cc:411: [4] (race) access:
  This usually indicates a security flaw. If an attacker can change anything
  along the path between the call to access() and the file's actual use
  (e.g., by moving files), the attacker can exploit the race condition
  (CWE-362/CWE-367). Set up the correct permissions (e.g., using setuid())
  and try to open the file directly.
src/leveldb/util/env_win.cc:582: [4] (format) vsnprintf:
  If format strings can be influenced by an attacker, they can be exploited,
  and note that sprintf variations do not always \0-terminate (CWE-134). Use
  a constant for the format specification.
src/leveldb/util/posix_logger.h:67: [4] (format) vsnprintf:
  If format strings can be influenced by an attacker, they can be exploited,
```

Рисунок 7 – результат работы flawfinder

## Вывод

В результате выполнения работы были изучены уязвимости C/C++. Для поиска уязвимостей существуют различные статические анализаторы исходного кода. Были изучены средства flawfinder, cppcheck, rats и clang. Одинаковый принцип работы и показатели имеют rats и flawfinder, поэтому их совместно использовать нет смысла, остальные же следует использовать параллельно.