|  |  |  |
| --- | --- | --- |
|  |  | УТВЕРЖДЕНО |
|  |  | Генеральный директор  ООО «Стандарты на века» |
|  |  | \_\_\_\_\_\_\_\_\_\_\_ А.М. Игнатьев |
|  |  | “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  |  |  |

Программный комплекс «Садовники»

**Протокол испытаний № 2**

643.00100101.11111-01 51 01-ЛУ

Листов 13

|  |  |  |
| --- | --- | --- |
|  |  | Ответственный исполнитель |
|  |  |  |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Киреев В.В. |
|  |  | “\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  |  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Санкт-Петербург

2024

ОГЛАВЛЕНИЕ

[1 Контроль состава и содержания документации 3](#_Toc188548576)

[2 Контроль исходного состояния ПО 3](#_Toc188548577)

[3 Контроль полноты и отсутствия избыточности исходных текстов контролируемого ПО на уровне файлов 4](#_Toc188548578)

[4 Контроль соответствия исходных текстов ПО его загрузочному коду 5](#_Toc188548579)

[5 Контроль связей функциональных объектов по управлению 5](#_Toc188548580)

[6 Контроль связей функциональных объектов по информации 6](#_Toc188548581)

[7 Контроль информационных объектов 6](#_Toc188548582)

[8 Формирования перечня маршрутов выполнения функциональных объектов 7](#_Toc188548583)

[9 Контроль выполнения функциональных объектов 9](#_Toc188548584)

[10 Сопоставления фактических маршрутов выполнения функциональных объектов и маршрутов, построенных в процессе проведения статического анализа 11](#_Toc188548585)

[11 Интегральная оценка проведённых испытаний 13](#_Toc188548586)

# 1 Контроль состава и содержания документации

Перечень программной документации, предоставленной Разработчиком при проведении испытаний:

1. Руководство по эксплуатации ПО «Садовники»:   
   643.00100102.11111-01 91 01.
2. Спецификация ПО «Садовники»: 643.00100102.11111-01 10 01.
3. Описание программы ПО «Садовники»: 643.00100102.11111-01 13 01.
4. Текст программы ПО «Садовники»: 643.00100102.11111-01 12 01.
5. Пояснительная записка ПО «Садовники»: 643.00100102.11111-01 81 01.

Данный перечень программной документации полностью совпадает с заявленным на проведение испытаний перечнем документов.

Проверка программной документации, предоставленной Разработчиком при проведении испытаний:

1. Спецификация ПО «Садовники»: 643.00100102.11111-01 10 01 соответствует требованиям к содержанию и оформлению ГОСТ 19.202-78.
2. Описание программы ПО «Садовники»: 643.00100102.11111-01 13 01 соответствует требованиям к содержанию и оформлению ГОСТ 19.402-78.
3. Текст программы ПО «Садовники»: 643.00100102.11111-01 12 01 соответствует требованиям к содержанию и оформлению ГОСТ 19.401-78.
4. Пояснительная записка ПО «Садовники»: 643.00100102.11111-01 81 01 соответствует требованиям к содержанию и оформлению ГОСТ 19.404-79.

В ходе испытаний было заключено, что состав и содержание полностью соответствует РД, на основе которого проводятся испытания.

**Испытание пройдено успешно.**

# 2 Контроль исходного состояния ПО

Были рассчитаны хэш-суммы представленного исходного программного продукта и хэш-суммы продукта, представленного на испытания (хеш-сумма SHA256 рассчитывается с помощью программного средства фиксации и контроля исходного состояния «hashtab»).

Расчет уникальных значений контрольных сумм загрузочных модулей и исходных текстов программ, входящих в состав ПО:

|  |  |  |  |
| --- | --- | --- | --- |
| Файл | Рассчитанное в ходе испытаний hash-значение | Hash-значение, указанное в описание программы | Эталонный образец разработчика |
| gardeners\_vuln.cpp | 4d81ba06fa682cab2fb25b515d270d7263b7f942fc2c0c7b3a29740de6e70588 | 4d81ba06fa682cab2fb25b515d270d7263b7f942fc2c0c7b3a29740de6e70588 | 4d81ba06fa682cab2fb25b515d270d7263b7f942fc2c0c7b3a29740de6e70588 |
| Gardeners\_vuln.exe | 049b02bb5c762c4d2ce4350349c1d18cd6b2317d08df86b1652de3c209573b6f | 049b02bb5c762c4d2ce4350349c1d18cd6b2317d08df86b1652de3c209573b6f | 049b02bb5c762c4d2ce4350349c1d18cd6b2317d08df86b1652de3c209573b6f |

Требуемые КС исследуемого ПО полностью совпадают:

**Испытание пройдено успешно.**

# 3 Контроль полноты и отсутствия избыточности исходных текстов контролируемого ПО на уровне файлов

Перечень исходных текстов (файлов), необходимых и достаточных для корректной сборки исследуемого ПО, предъявляемый заявителем:

|  |  |
| --- | --- |
| **Дистрибутив** | **Исходные файлы** |
| gardeners\_vulnerable | gardeners\_vuln.cpp |
| gardeners\_vulnerable | gardeners\_vulnerable.exe |

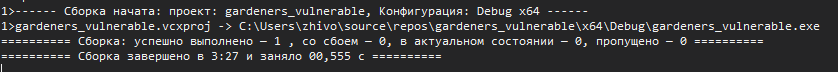


Рис.1 Сборка всех файлов и компиляция ПО

Полнота представленных исходных текстов на уровне файлов ПО подтверждается фактом успешной компиляции и сборки исследуемого ПО.

Избыточность представленных исходных текстов на уровне файлов ПО подтверждается фактом отсутствия файлов, которые не используются при выполнении функциональных требований ПО.

**Испытание пройдено успешно.**

# 4 Контроль соответствия исходных текстов ПО его загрузочному коду

|  |  |  |
| --- | --- | --- |
| **Файл** | **Данные о КС, предъявляемые заявителем** | **Контроль, проведённый в рамках текущего испытания** |
| gardeners\_vulnerable.exe | 049b02bb5c762c4d2ce4350349c1d18cd6b2317d08df86b1652de3c209573b6f | 049b02bb5c762c4d2ce4350349c1d18cd6b2317d08df86b1652de3c209573b6f |

Требуемые КС исследуемого ПО полностью совпадают:

**Испытание пройдено успешно.**

# 5 Контроль связей функциональных объектов по управлению

Ниже приведена таблица связи функциональных объектов: если функциональный объект осуществляет вызов другого функционального объекта, то поле пересечения обозначается символом «+», где столбец соответствует функциональному объекту, который вызывает другой функциональный объект, а строка – вызываемому объекту.

Таблица 1. Связи функциональных объектов по управлению

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | void print\_array | DWORD WINAPI FirstGardener | DWORD WINAPI SecondGardener | int main |
| void print\_array |  | + | + |  |
| DWORD WINAPI FirstGardener | + |  |  | + |
| DWORD WINAPI SecondGardener | + |  |  | + |
| int main |  | + | + |  |

Все установленные связи функциональных объектов по управлению полностью соответствуют логике работы тестового ПО, заявленного в документации (пояснительной записке).

**Испытание пройдено успешно.**

# 6 Контроль связей функциональных объектов по информации

Таблица 2. Связи функциональных объектов по информации

|  |  |  |
| --- | --- | --- |
| **Функциональный Объект** | **Имя ИО** | **Тип ИО** |
| main  print\_array | cs | CRITICAL\_SECTION |
| print\_array | array | vector[vector[]] |
| main | gardener1, gardener2 | HANDLE |
| FirstGardener -> print\_array  SecondGardener -> print\_array | point, i, j | int |

Исходя из анализа связей функциональных объектов по информации, установлено, что глобальные и локальные переменные соответствуют логике работы тестового ПО, заявленной в документации (пояснительной записке), а также не нарушает принципы безопасности критических переменных.

**Испытание пройдено успешно.**

# 7 Контроль информационных объектов

Таблица 3. Информационные объекты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя ИО** | **Описание** | **Тип ИО** | **Граничные значения** | **Область видимости** |
| cs | Объект критической секции | СRITICAL SECTION | - | Глобальная |
| array | Двумерный массив — сад для садовников | std::vector<std::vector<int>> | 0 – зависит от доступной памяти | Глобальная |
| gardener1, gardener2 | Хендлеры для потоков | HANDLE | - | Локальные |
| point, i, j | Местоположение садовников | int | 4 байта (-2,147,483,648 - 2,147,483,647) | Локальные |

# 8 Формирования перечня маршрутов выполнения функциональных объектов

На основе предыдущих пунктов был построен граф, отражающий маршруты выполнения функциональных объектов.

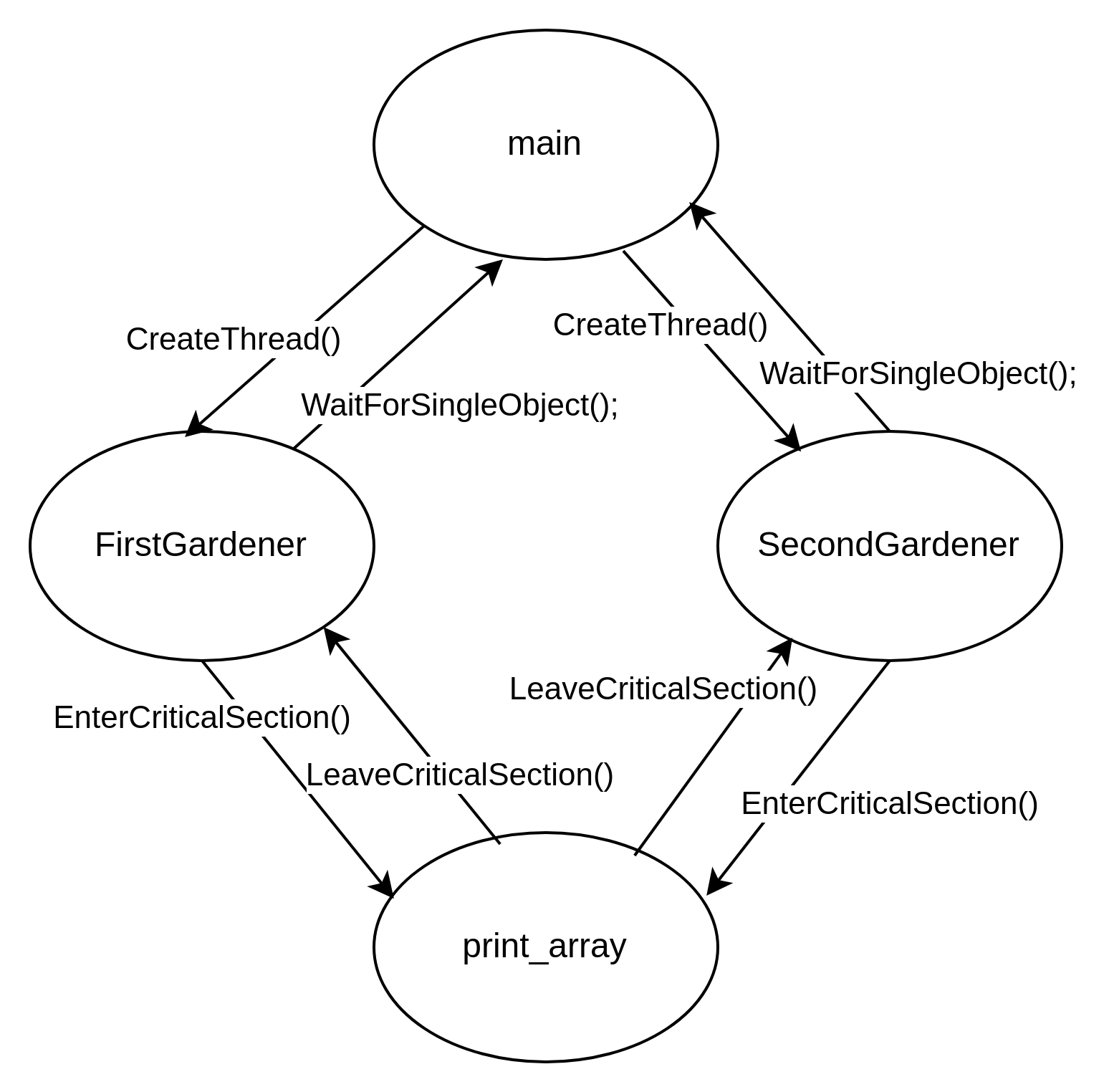


Рис.3 Граф маршрутов трасс ПО

В ходе анализа маршрутов выполнения функциональных объектов было заключено, что статические маршруты вызовов функциональных объектов, а также их порядок и последовательность соответствуют логике работы тестового ПО, заявленной в документации (пояснительной записке).

В процессе тестирования программного обеспечения с помощью статического анализатора PVS-studio была выявлена уязвимость, связанная с переполнением буфера (CWE-119: Выход операций за границы буфера в памяти). Эта уязвимость возникает в функции, отвечающей за заполнение фиксированного массива символов, где не осуществляется проверка границ при записи данных. В результате, при попытке записать данные в массив, программа может записать за пределами выделенной памяти, что приводит к повреждению данных в памяти.

Наличие критических уязвимостей свидетельствует о провале проверки на наличие НДВ.

**Испытание не пройдено.**

# 9 Контроль выполнения функциональных объектов

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемое действие** | **Ожидаемый результат** | **Полученный результат** | **Оценка** |
| Садовник 1 засаживает одну клетку сада | Садовник 1 засаживает одну клетку сада | Садовник 1 засаживает одну клетку сада | + |
| Садовник 2 засаживает одну клетку сада | Садовник 2 засаживает одну клетку сада | Садовник 2 засаживает одну клетку сада | + |
| Садовник 2 ждёт пока садовник 1 засадит клетку сада | Когда садовник 1 засаживает клетку, садовник 2 не засаживает свою клетку | Садовник 1 засаживает клетку, садовник 2 не засаживает свою клетку | + |
| Садовник 1 ждёт пока садовник 2 засадит клетку сада | Когда садовник 2 засаживает клетку, садовник 1 не засаживает свою клетку | Садовник 2 засаживает клетку, садовник 1 не засаживает свою клетку | + |

Датчики:

std::cout << "Main called\n";

std::cout << "Main exited\n";

std::cout << "FirstGardener thread intialized\n";

std::cout << "FirstGardener thread has done its' work\n";

std::cout << "SecondGardener thread intialized\n";

std::cout << "SecondGardener thread has done its' work\n";

std::cout << "First gardener started printing array (first thread using shared resource)\n";

std::cout << "Second gardener started printing array (second thread using shared resource)\n";

std::cout << "First gardener stopped printing array (first thread stopped using shared resource)\n";

std::cout << "Second gardener stopped printing array (second thread stopped using shared resource)\n";

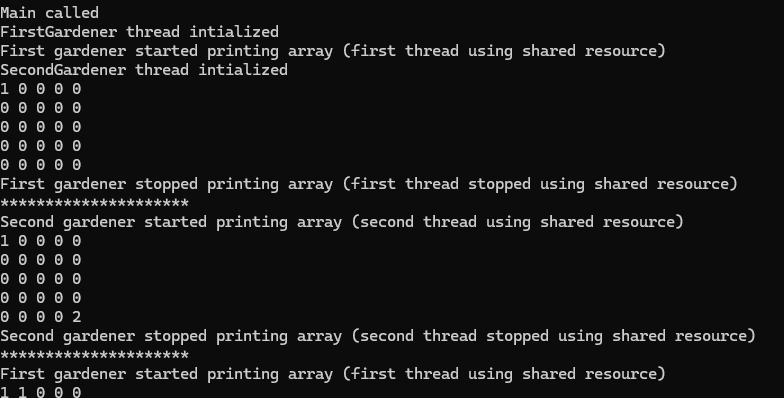


Рисунок 3 – Проверка выполнения функциональных объектов

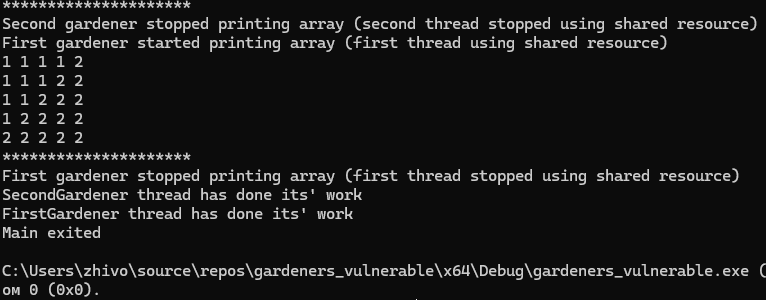


Рисунок 4 – Проверка выполнения функциональных объектов

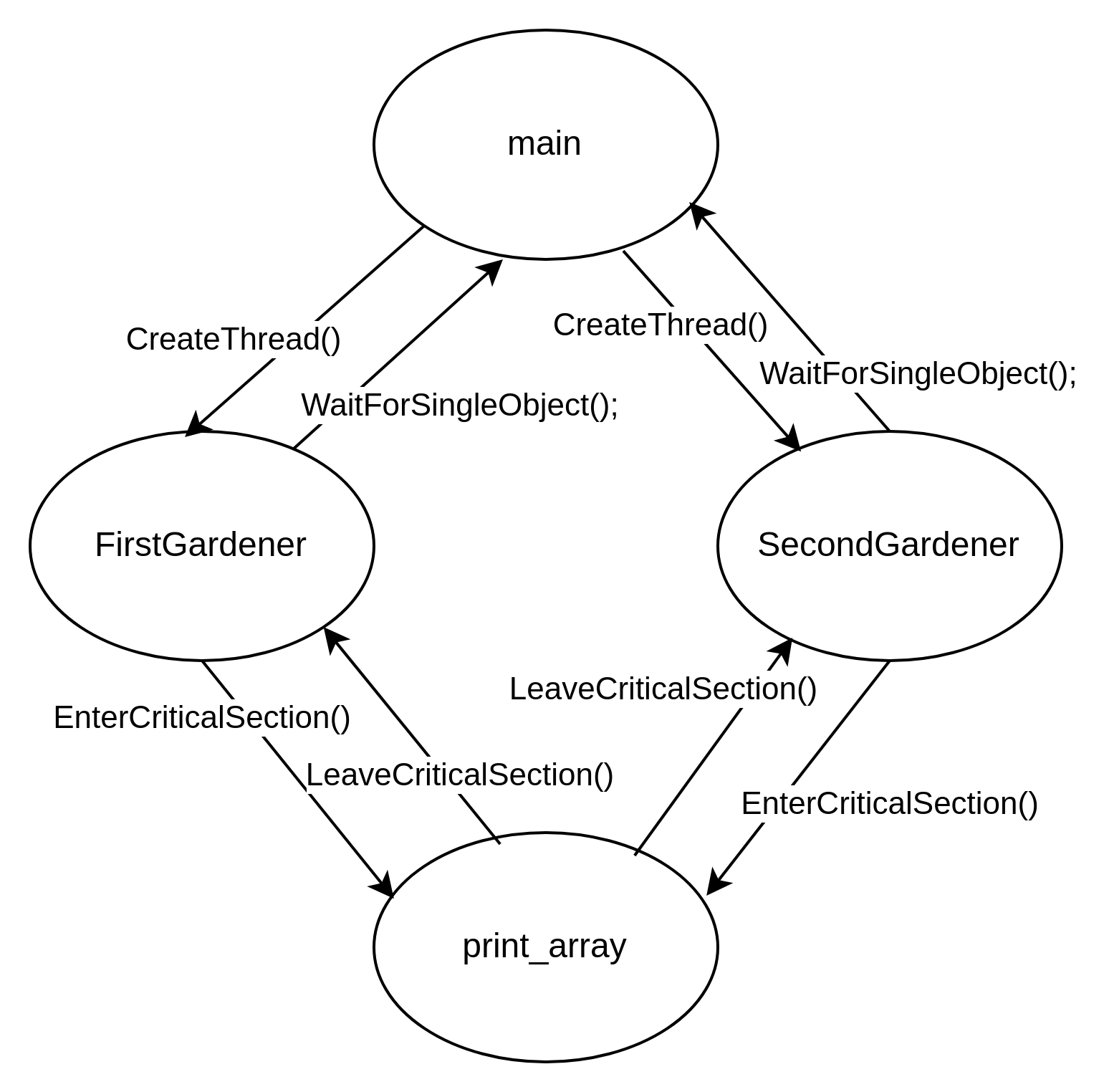
В процессе тестирования программного обеспечения была выявлена уязвимость, связанная с переполнением буфера (CWE-119: Выход операций за границы буфера в памяти). Эта уязвимость возникает в функции, отвечающей за заполнение фиксированного массива символов, где не осуществляется проверка границ при записи данных. В результате, при попытке записать данные в массив, программа может записать за пределами выделенной памяти, что приводит к повреждению данных в памяти.

Несмотря на то, что программа не завершает работу аварийно, переполнение буфера может привести к непредсказуемому поведению, включая случайные сбои в работе других функциональных блоков. Например, если данные, записанные за пределами массива, затрагивают другие переменные или структуры данных, это может привести к неправильной работе программы, сбоям в логике выполнения и даже к уязвимостям безопасности, позволяющим злоумышленникам выполнять произвольный код.

**Испытание не пройдено.**

# 10 Сопоставления фактических маршрутов выполнения функциональных объектов и маршрутов, построенных в процессе проведения статического анализа

**На основании п.8 и п.9 установлено, что построенные фактические маршруты выполнения соответствуют маршрутам, построенным в процессе проведения статического анализа.**



**Рис.4 – Граф маршрутов**

**Однако в ходе тестирования была выявлена уязвимость, связанная с переполнением буфера (CWE-119: Выход операций за границы буфера в памяти). Эта уязвимость возникает в функции, отвечающей за заполнение фиксированного массива символов, где не осуществляется проверка границ при записи данных. В результате, при попытке записать данные в массив, программа может записать за пределами выделенной памяти, что приводит к повреждению данных.**

**Несмотря на то, что фактические маршруты выполнения функций совпадают с маршрутами, построенными в процессе статического анализа, некорректные данные, вызванные переполнением буфера, могут привести к сбоям в работе других функциональных блоков. Это делает невозможным корректное отображение состояния системы и может привести к неправильной работе программы.**

**Испытание не пройдено.**

# 11 Интегральная оценка проведённых испытаний

|  |  |
| --- | --- |
| **Испытание** | **Результат** |
| Контроль состава и содержания документации | успешно |
| Контроля исходного состояния ПО | успешно |
| Контроля полноты и отсутствия избыточности исходных текстов контролируемого ПО на уровне файлов | успешно |
| Контроля соответствия исходных текстов ПО его загрузочному коду. | успешно |
| Контроля полноты и отсутствия избыточности исходных текстов контролируемого ПО на уровне функциональных объектов (процедур) | успешно |
| Контроля связей функциональных объектов по управлению | успешно |
| Контроля связей функциональных объектов по информации | успешно |
| Контроля информационных объектов различных типов (например, локальных переменных, глобальных переменных, внешних переменных и т.п.) | успешно |
| Формирования перечня маршрутов выполнения функциональных объектов | успешно |
| Контроля выполнения функциональных объектов | не успешно |
| Сопоставления фактических маршрутов выполнения функциональных объектов (процедур, функций) и маршрутов, построенных в процессе проведения статического анализа. | не успешно |

Некоторые испытания ПО «Садовники» на контроль отсутствия НДВ согласно РД не пройдены. Исследуемое ПО не может соответствовать 3-му уровню контроля отсутствия НДВ согласно РД «Защита от НСД к информации. Часть 1. ПО СЗИ. Классификация по уровню контроля отсутствия НДВ».