## Слайд 1

Добрый день, уважаемые председатель и члены государственной аттестационной комиссии! Вашему вниманию представляется ВКР по разработке программного модуля для анализа программ на языках C и C++ на недекларированные возможности.

Целью данной работы было ускорение проведения процесса сертификации программ, написанных на языках C/C++

## Слайд 2

На текущий момент проверка программ на НДВ органом сертификации происходит вручную:

- с помощью специального ПО проводят статический и динамический анализ исходных кодов программного проекта и скомпилированной программы соответственно;
- результаты анализов приводятся к общему виду;
- резрабатывается скрипт для сравнения результатов;
- с помощью скрипта сравнения ищутся несовпадения.

В свою очередь, ПМ АПНДВ умеет автоматически проводить действия, требуемые для проверки ПО, возращая оператору отчет, на основе которого оператор ПМ АПНДВ делает заключение о наличии или отсутствии НДВ в тестируемом ПО.

## Слайд 3

Сейчас в открытом доступе не существует программных решений, аналогичных ПМ АПНДВ, поэтому для ПМ АПНДВ сравнивался функционал статических и динамических анализаторов, а наиболее подходящее по критериям ПО становилось частью ПМ АПНДВ.

В качестве статического анализатора была выбрана программа GNU Cflow как наиболее подходящая по всем критериям.

## Слайд 4

В качестве статического анализатора был выбран отладчик GDB как более быстрая и расширяемая альтернатива эмулятору Qemu.

## Слайд 5

В качестве языка программирования был выбран Nim, как язык с хорошим балансом между скоростью работы, легкостью написания кода и портируемостью. Nim компилируется в машинный код, через промежуточную компиляцию в код Си, что позволяет ему пользоваться всей стабильностью и высоким уровнем оптимизации, который предоставляют такие компиляторы, как gcc или clang.

## Слайд 6

В качестве среды разработки был выбран текстовый редактор Vim. Vim не требователен к ресурсам, в отличие от большинства других популярных редакторов кода и IDE, имеет большую библиотеку плагинов, позволяющих идеально настроить окружение под себя.

## Слайд 7

Схема данных отражает операции, происходящие с данными во время работы ПМ АПНДВ. От оператора программа получает папку с мейкфайлом и именем программы, после чего собирает программу из исходников, проводит статический, динамический и сравнительный анализ, выводя оператору отчет о проделанной работе.

## Слайд 8

Схема алгоритма отражает саму работу программы. Первым идет этап сборки, от которого зависит как статический, так и динамический анализ. После окончания сборки параллельно запускаются этапы статического и динамического анализа, а по завершению их обоих проходит сравнительный анализ.

В статическом анализе строится дерево вызовов программы, основываясь на том, как вызовы записаны в исходниках. После чего дерево переводится во внутренний формат, и к нему добавляется информация времени компиляции, связанная с адресами функций в исполняемом файле.

Этап динамического анализа предваряет бинарный анализ, в котором ищется первая инструкция в сегменте кода программы и на нее выставляется программная точка останова. После этого ПМ АПНДВ собирает динамические трассы программы, вместе с мета-информацией о вызовах, и последним шагом преобразует собранную информацию во внутренний формат хранения данных.

## Слайд 9

Графический пользовательский интерфейс минималистичен и отражает собой простоту запуска программы.

## Слайд 10

Разработка ПМ АПНДВ проводилась с использованием техники Test-driven development, которая предусматривает написание тестов для кода перед тем, как разработчик напишет сам программный код, что помогает продумывать интерфейсы взаимодействия между элементами программы. В рамках данной техники было написано 43 модульных теста.

## Слайд 11

Внедрение ПМ АПНДВ готовится в ООО Фирма “Анкад”.