ИСП РАН

Программное обеспечение «АвтоФАЗЗ»

Краткое руководство пользователя

2026 г.

Содержание

1. Описание функциональных возможностей

Инструмент «АвтоФАЗЗ» предназначен для автоматизированного внедрения фаззинг-тестирования в CI-процессы существующих проектов. Настоящий документ является кратким руководством пользователя Инструмента «АвтоФАЗЗ».

Инструмент позволяет автоматически создавать фаззинг-цели на основе инкрементальных снимков состояния программы и формировать корпус начальных входных данных в процессе выполнения программы на основе существующих тестов.

К основным функциям Инструмента «АвтоФАЗЗ» относятся:

- добавление задачи фаззинг-тестирования в существующие CI-конвейеры;
- автоматическое создание фаззинг-целей;
- автоматическое формирование корпуса начальных входных данных.

Минимальные системные требования: для функционирования Инструмента «АвтоФАЗЗ» требуется операционная система Ubuntu 20.04, процессор архитектуры x86-64 (не менее 4 ядер), 32 ГБ оперативной памяти, размер свободного дискового пространства не менее 100 ГБ.

2. Лицензирование и установка

По вопросам определения стоимости, приобретения и использования обращайтесь по адресу: ISP_AutoFuzz@ispras.ru.

2.1 Системные требования

Минимальные системные требования: Для функционирования Инструмента требуется персональный компьютер с архитектурой x86-64 на базе процессора Intel, не менее 32 ГБ оперативной памяти, размер свободного дискового пространства не менее 100 ГБ. Поддерживаются следующие 64-разрядные дистрибутивы операционной системы Linux: Ubuntu 20.04.

Список используемого ПО с указанием лицензий.

№	Название ПО	Лицензия	Открытый репозиторий
1	libnyx	GNU General Public License v2.0	https://github.com/nyx-fuzz/libnyx
2	packer	GNU General Public License v2.0	https://github.com/nyx-fuzz/packer
3	QEMU-Nyx	GPL v2.0	https://github.com/nyx-fuzz/QEMU-Nyx
4	KVM-Nyx	GPL v2.0 (для патчей)	https://github.com/nyx-fuzz/KVM-Nyx
5	libxdc	MIT License	https://github.com/nyx-fuzz/libxdc

2.2 Подготовка окружения хост-системы

2.2.1 Установка модуля KVM-NYX

На хост-системе необходимо установить ядро Linux версии 5.10.75 для Ubuntu и соответствующие модули:

```
cd /tmp/ &&
wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.10.75/amd64/linux-image-unsigned-5.10.75-051075-generic_5.10.75-051075.202110201038_amd64.deb &&
wget -c https://kernel.ubuntu.com/~kernel-ppa/mainline/v5.10.75/amd64/linux-modules-5.10.75-051075-generic_5.10.75-051075.202110201038_amd64.deb &&
sudo dpkg -i *.deb
sudo reboot
```

Далее необходимо загрузить систему с новым ядром и установить KVM-NYX. Для этого необходимо:

```
# Удалить стандартные модули KVM
sudo rmmod kvm_intel kvm
# Установить модули KVM-NYX
sudo insmod ./kvm.ko
```

```
sudo insmod ./kvm-intel.ko
sudo chmod 777 /dev/kvm
Установить зависимости:
sudo apt install -y libvdeplug-dev
```

Примечание - После каждой перезагрузки системы с NYX-ядром необходимо повторять команды:

```
sudo rmmod kvm_intel kvm
sudo insmod ./kvm.ko
sudo insmod ./kvm-intel.ko
sudo chmod 777 /dev/kvm
```

Для корректной работы АвтоФАЗЗ рекомендуется создать в корневом каталоге директорию `/autofuzz`:

```
sudo mkdir -p /autofuzz
```

Если в корневом разделе недостаточно свободного места, рекомендуется разместить файлы в другой директории и выполнить bind-монтирование в `/autofuzz`:

```
cd path/to/autofuzz
sudo mount --bind $(pwd) /autofuzz
```

2.2.2 Подготовка инструментов

Инструменты следует разместить по следующим путям:

1. `/autofuzz/crusher` - релиз комплекса гибридного анализа Crusher
2. `/autofuzz/crusher/nyx_mode/packer/packer/nyx.ini`

Для этого необходимо скопировать файл:

```
cp /path/to/crusher/nyx_mode/autofuzz/nyx_files/nyx.ini
/autofuzz/crusher/nyx_mode/packer/packer/
```

3. `/autofuzz/crusher/nyx_mode/packer/packer/fuzzer_configs/default_config_vm.ron`

Для этого необходимо скопировать файл:

```
cp /path/to/crusher/nyx_mode/autofuzz/nyx_files/default_config_vm.ron
/autofuzz/crusher/nyx_mode/packer/packer/fuzzer_configs/
```

4. `/autofuzz/tool`

Для этого необходимо скопировать директорию:

```
cp -r /path/to/crusher/nyx_mode/autofuzz/tool /autofuzz/
```

После копирования в директории `/autofuzz/tool` необходимо выполнить скрипт:

```
./setup.sh
```

5. `/autofuzz/base_vm.img` - образ виртуальной машины (см. раздел 2.2.3)
6. `/autofuzz/work` - необходимо создать пустую директорию:

```
mkdir -p /autofuzz/work
```

2.2.3 Подготовка виртуальной машины

Необходимо подготовить образ виртуальной машины, который должен соответствовать следующим требованиям:

1. В качестве образа виртуальной машины следует использовать RAW-образ диска QEMU, внутри которого возможно запускать тестирование целевого программного обеспечения.
2. Внутри виртуальной машины должен быть создан пользователь `user` с правом `sudo` без ввода пароля.
3. Должен быть обеспечен SSH-доступ (порт 22) к пользователю `user` без ввода пароля (по ключу).
4. Внутри виртуальной машины должны присутствовать скрипты `/root/root_on_start.sh` и `/home/user/on_start.sh` (их необходимо скопировать из директории `/crusher/nyx_mode/autofuzz/in_vm`).
5. При включении виртуальной машины должен автоматически запускаться скрипт `/root/root_on_start.sh`.
6. Внутри виртуальной машины должна существовать директория `/home/user/work`.

2.3 Настройка Jenkins

Предполагается, что у пользователя уже настроены CI-задания сборки и тестирования целевого программного обеспечения.

Например, скрипт сборки может иметь следующий вид:

```
#!/usr/bin/env bash
git clone https://.../project.git
cd project
./configure
make -j
zip -r build.zip bin_dir/ # artifact
```

Пример скрипта тестирования:

```
#!/usr/bin/env bash
cd project
make test
```

2.3.1 Добавление узла `autofuzz`

Для добавления узла `autofuzz` необходимо перейти в веб-интерфейс Jenkins. Для добавления CI-задачи фаззинг-тестирования необходимо создать CI-узел на основе подготовленной в разделе 2.2 хост-системы.

На хосте должен быть предусмотрен пользователь `jenkins`, состоящий в группе `kvm` (для запуска QEMU в режиме KVM) и имеющий доступ к файлам директории `/autofuzz` (например, владелец этих файлов).

Jenkins должен иметь доступ по SSH к этому пользователю, для этого необходимо добавить ключ Jenkins-сервера в `~/.ssh/authorized_keys`.

На хосте должна быть установлена Java.

При добавлении узла для АвтоФАЗЗ следует настроить параметры следующим образом:

- Название узла - `autofuzz`
- Количество одновременных исполнителей - `1`
- Корневая директория - `/home/jenkins`
- Использование - `Собирать только проекты с метками, совпадающими с этим узлом`
- Способ запуска - `Launch agents via SSH (запускать агентов по SSH)` и указать сетевой адрес хоста
- Переменные среды (список пар “ключ-значение”):
  - `AUTOFUZZ_ROOT_DIR = /autofuzz`
  - `PATH+MY_BIN = /autofuzz/tool/exe`

2.3.2 Настройка задания сборки

В параметрах задания сборки необходимо в полях выставить следующие значения:

- Ограничить лейблы сборщиков, которые могут исполнять данную задачу (Label Expression) - `autofuzz`
- Inject environment variables to the build process (Properties Content) - `AUTOFUZZ_ON=1` (включить АвтоФАЗЗ)

Использование узла `autofuzz` для сборки позволяет избежать проблем при переносе скомпилированных файлов в тестовое окружение.

2.3.3 Настройка задания тестирования

В параметрах задания тестирования необходимо в полях выставить следующие значения:

- Ограничить лейблы сборщиков, которые могут исполнять данную задачу (Label Expression) - `autofuzz`
- Inject environment variables to the build process (Properties Content) –
  - `AUTOFUZZ_ON=1` (включить АвтоФАЗЗ)
  - `AUTOFUZZ_MAKE_SNAPSHOTS=1` - сохранять снимки состояния для фаззинга на точках чтения входных данных
  - `AUTOFUZZ_TARGET_EXECUTABLE="my_exe"` - название исполняемого файла целевого ПО (делать снимки только на точках чтения в нём)
  - `AUTOFUZZ_TIME_JOB=5M` - ограничение времени (суффикс H/M/S - часы/минуты/секунды)

2.3.4 Создание задания фаззинга

Далее необходимо создать задание фаззинга (`+ New Item`).

В параметрах задания необходимо указать узел `autofuzz`. В шаге `Execute shell` после строки `#!/usr/bin/env bash` можно не указывать дополнительные команды. Также необходимо выставить переменные среды (первые две - обязательные, остальные - опционально):

- `AUTOFUZZ_ON=1` - включить АвтоФАЗЗ
- `AUTOFUZZ_RUN_FUZZING=1` - запустить фаззинг (используя ранее созданные снимки)
- `AUTOFUZZ_NUM_SNAP=5` - сколько снимков фаззить
- `AUTOFUZZ_TIME_SNAP=1H` - сколько по времени фаззить каждый снимок (суффикс H/M/S - часы/минуты/секунды)
- `AUTOFUZZ_EXEC_TIME_MS=5000` - ограничение времени одного запуска в процессе фаззинга (в миллисекундах)

В результате для каждого снимка из указанного множества будет запущен фаззинг на указанное время, после чего директории с результатами фаззинг-тестирования могут быть скачаны в составе артефактов задания.

Рекомендуется ограничивать количество снимков, поскольку их число может быть слишком велико.

3. Сценарий работы с Инструментом «АвтоФАЗЗ»

После настройки Инструмента в соответствии с разделом 2 настоящего руководства типовой сценарий работы с Инструментом «АвтоФАЗЗ» включает следующие этапы:

3.1. Запуск сборки проекта

На первом этапе пользователь запускает CI-задание сборки целевого проекта. В результате выполнения данного этапа формируются артефакты сборки, необходимые для последующих этапов работы.

3.2. Запуск тестирования и создание снимков

После успешного завершения сборки необходимо запустить CI-задание тестирования. На данном этапе выполняются тесты целевого программного обеспечения, создаются инкрементальные снимки состояния программы или виртуальной машины в точках чтения входных данных, а также формируется начальный корпус входных данных для последующего фаззинг-тестирования.

3.3. Запуск фаззинг-тестирования

После завершения этапа тестирования пользователь запускает CI-задание фаззинг-тестирования, добавленное в соответствии с разделом 2 настоящего руководства. На данном этапе Инструмент «АвтоФАЗЗ» использует ранее сформированные снимки и начальный корпус входных данных для автоматического запуска фаззинг-тестирования для сгенерированных фаззинг-целей.

3.4. Получение и анализ результатов

По завершении фаззинг-тестирования пользователю становятся доступны артефакты, содержащие результаты выполнения.

3.5. Повторный запуск процесса

При необходимости пользователь может повторно выполнить описанный сценарий для новой версии целевого проекта или после изменения параметров CI-конвейера. Рекомендуется выполнять этапы в следующей последовательности: сборка, тестирование, фаззинг-тестирование.

4. Описание модулей Инструмента «АвтоФАЗЗ»

4.1. Модуль создания инкрементальных снимков

Обеспечивает создание инкрементальных снимков состояния виртуальной машины в точках выполнения операций чтения входных данных, организацию снимков в виде иерархического дерева состояний и их сериализацию в файловую систему во время выполнения CI-задания тестирования.

4.2. Модуль генерации фаззинг-целей

Обеспечивает автоматическое формирование фаззинг-целей на основе инкрементальных снимков состояния программы.

4.3. Модуль формирования начального корпуса входных данных

Обеспечивает формирование начального корпуса входных данных на основе существующих тестов целевого ПО, включая извлечение входных данных из перехваченных операций чтения.

4.4. Модуль проведения фаззинг-тестирования

Обеспечивает запуск фаззинг-тестирования для каждой сгенерированной фаззинг-цели с использованием соответствующих инкрементальных снимков и начального корпуса входных данных во время выполнения CI-задания фаззинга.

4.5. Модуль сохранения результатов

Обеспечивает формирование артефактов сборки, тестирования и фаззинг-тестирования.

5. Описание процессов, обеспечивающих поддержание жизненного цикла ПО

5.1 Процессы разработки и совершенствования ПО

Разработка Инструмента «АвтоФАЗЗ», предназначенного для автоматизированного внедрения фаззинг-тестирования в CI-процессы, ведется по методологии Agile с привлечением современных средств повышения качества кода.

1. Для хранения кода используется система контроля версий Git, и изменения в основной ветке должны пройти все тесты, а также инспекцию кода (code review) другими разработчиками.
2. При разработке используется практика непрерывной интеграции (continuous integration): при помощи CI/CD-системы GitLab происходят регулярные автоматические сборки с последующим запуском тестов, упомянутых в предыдущем пункте.
3. При написании кода разработчики должны придерживаться строгих правил оформления кода. Нарушение этих правил не позволит пройти этап инспекции кода.

5.2 Поддержка пользователей ПО

Развертывание Инструмента «АвтоФАЗЗ» требует предварительной подготовки программного окружения, включая установку необходимых компонентов и настройку среды выполнения в соответствии с настоящим руководством. Для функционирования инструмента необходимо наличие предустановленных компонентов: ПС Crusher и системы непрерывной интеграции (CI/CD).

Со своей стороны, ИСП РАН постоянно совершенствует Инструмент «АвтоФАЗЗ», применяя в жизненном цикле разработки передовые методики. Добавление новых и улучшение существующих алгоритмов ведется в инициативном порядке. Обновления Инструмента «АвтоФАЗЗ» передаются пользователям через согласованные с ними каналы распространения обновлений.

5.3 Необходимый персонал для разработки и поддержки

Для разработки и поддержки программного продукта необходима соответствующая квалификация разработчиков. Это обусловлено следующей причиной:

- специфическая предметная область, требующая глубоких знаний одновременно в нескольких областях: динамический анализ кода, фаззинг-тестирование, виртуализация и работа с гипервизорами.

В силу приведенных причин коллектив разработчиков Инструмента «АвтоФАЗЗ» формируется из специалистов, получивших высшее профильное образование.

Для гарантийного обслуживания задействовано два научных сотрудника, для Технической поддержки задействован один научный сотрудник, для модернизации программного обеспечения задействованы два научных сотрудника.

Адрес электронной почты, по которому можно обратиться по вопросам, связанным с Инструментом «АвтоФАЗЗ» — ISP_AutoFuzz@ispras.ru.

6. Перечень терминов и сокращений

Система	Инструмент «АвтоФАЗ»
CI/CD	Непрерывная интеграция и непрерывная доставка
QEMU	Система эмуляции и виртуализации
KVM	Модуль виртуализации ядра Linux
KVM-NYX	Модифицированный модуль KVM из проекта Nyx
SSH	Протокол защищённого удалённого доступа
RAW-образ	Образ диска виртуальной машины в формате RAW