NAME

PMFuzz

SYNOPSIS

PMFuzz [-h] [--force-resp FORCE_RESP] [--cores-stage1 CORES_STAGE1] [--cores-stage2 CORES_STAGE2] [--overwrite] [--disable-stage2] [--dry-run] [--progress-interval PROGRESS_INTERVAL] [--progress-file PROGRESS_FILE] [--checks-only] [--verbose] [--version] indir outdir config

DESCRIPTION

PMFuzz: A Persistent Memory Fuzzer, version 1.0 by authors

OPTIONS

indir path to directory containing initial test corpus for stage 1. See *INPUT−DIRECTORY*.

outdir path to directory for generated test cases for stage 1, works as input for stage 2. See *OUTPUT−DIRECTORY*.

config Points to the config file to use, should conform to: configs/default.yaml. See *CONFIGURATION−FILE*.

−−force−resp *FORCE_RESP*
Forces response to questions

−−cores−stage1 *CORES_STAGE1*, **−c1** *CORES_STAGE1*
Maximum cores stage 1 fuzzer can use, default: 1. Can be specified in config.

−−cores−stage2 *CORES_STAGE2*, **−c2** *CORES_STAGE2*
Maximum cores stage 2 fuzzer can use, default: 1. Can be specified in config.

−−overwrite, **−o**
Overwrite the output directory

−−disable−stage2, **−1**
Disables stage 2. Can be specified in config.

−−dry−run
Enables dry run, no actual commands are executed (Deprecated)

−−progress−interval *PROGRESS_INTERVAL*
Interval in seconds for recording progress, default: 60 seconds. Can be specified in config.

−−progress−file *PROGRESS_FILE*
Output file for writing progress to a file. Can be specified in config.

−−checks−only
Performs startup checks and exits

−−verbose, **−v**
Enables verbose logging to stdout

−−version
show program's version number and exit

EXAMPLES

To run PMFuzz without any additional output:

```
pmfuzz−fuzz \
  ./input_directory ./output_directory ./configs/default.yml
```

To record fuzzing progress, use:

```
pmfuzz−fuzz \
  −−progress−file=/tmp/fuzz_progress
  ./input_directory ./output_directory ./configs/default.yml
```

To get verbose output of the PMFuzz´s current step, use:

```
pmfuzz−fuzz \
  −−verbose \
  ./input_directory ./output_directory ./configs/default.yml
```

INPUT DIRECTORY

TODO

OUTPUT DIRECTORY

TODO

CONFIGURATION FILES

PMFuzz uses a YAML based file to configure different parameters.

configs/examples directory contains several examples for writing and organizing configurations that PMFuzz can use. If you want to write your own configuration file, include **config/default.yml** in your new config file and change the values you need.

If you are writing your own configuration, please note the following:

Including Other Configs

PMFuzz supports including one or more configuration files to allow easier customization.

Syntax for including config files is:

```
include:
 − base−config−1.yaml
 − base−config−2.yaml
 .
 .
 − base−config−n.yaml
```

In case of duplicate keys, values are prioritized (and overwritten) in the order they appear. However, the file including them have highest priority.

Note
Nested includes are not supported.

Variable Substitution

The following variables are automatically substituted in the config file values:

%ROOT%
Points to the PMFuzz root directory (root of this repository)

%BUILD%
Points to the %ROOT%/build/

%LIB%
Points to the %ROOT%/build/lib

%BIN%
Points to the %ROOT%/build/bin

Example

Here is a simple example that runs PMDK´s RBTree workload in baseline mode. This configuration overwrites the number of CPU cores used by the first stage to 4. Note, lines starting with **#** are comments.

```
# Brief:
#   Runs the Baseline for rbtree

include:
  − configs/base.yml
  − configs/workload/mapcli.rbtree.yml
  − configs/run_configs/baseline.yml

pmfuzz:
  stage:
    "1":
      cores: 4
```

ENVIRONMENT

This section defines several environment variables that may change PMFuzz´s behavior.

Values **set** and **unset** describe the behavior when the environment variable is not set to any value and when the variable is set to any non−empty string (including **0**) respectively.

USE_FAKE_MMAP

1
Enables fake mmap by copying the contents (using **memcpy**) of the pool image to the volatile memory. Mounting the pool to the volatile memory improves fuzzing performance.

0
Mounts the pool using PMDK´s default mounting functions. Before invoking the target, PMFuzz would create a copy of the pool image and call the target on that image. Depending on the output of the fuzzing,PMFuzz would either save that image for future use, or discard it.

PMEM_MMAP_HINT

addr
Address of the mount point of the pool. See libpmem(7).

unset
PMDK decides the mount address of the pool.

ENABLE_CNST_IMG

1
Disables default PMDK´s behaviour that generates non−identical images for same input.

0
PMFuzz generated images would have random variations that may negatively affect the fuzzing

performance and reproducibility.

FI_MODE

"IMG_GEN"
In case the the PMFUZZ_MODE env variable is set to "IMG_GEN", a failure point is injected and the PM Image is dumped if the PM pool has changed since the last failure injection. First failure injection always results in an image dump.

Images dump naming pattern: **<pool file name>.<failure id>** If a failure list file is additionally specified using the env variable, the falure ids that generate dumps are written to that file, one per line.

"IMG_REP"
Todo

For more information on FI_MODE see libpmfuzz.c.

FAILURE_LIST
Path to a file that libpmfuzz would write the failure IDs to.

See libpmfuzz.c

PMFUZZ_DEBUG

1
Enables debug output from libpmfuzz.

0

unset
Disables debug output from libpmfuzz.

ENABLE_PM_PATH
Enables deep paths in PMFuzz

GEN_ALL_CS
TODO

IMG_CREAT_FINJ
TODO

PMFUZZ_SKIP_TC_CHECK

set
Disable testcase size check in AFL++.

unset
Enables AFL++´s default behaviour to check testcase size.

See afl−fuzz(1).

PRIMITIVE_BASELINE_MODE

set
Makes workload delete image on start if the pool exists

BUGS

Report bugs at: https://github.com/sihangliu/pmfuzz/issues

AUTHORS

PMFuzz was written by ShiftLab. Contact: <suyash12mahar@outlook.com>.

DISTRIBUTION

The latest version of PMFuzz may be downloaded from ⟨https://pagure.io/myproject⟩