An Empirical Study on the Distance Metric in Guiding Directed Grey-box Fuzzing (Paper Artifact)

This is the artifact of the paper *An Empirical Study on the Distance Metric in Guiding Directed Grey-box Fuzzing* to appear in ISSRE 2024.

DOI 10.5281/zenodo.13369424

https://github.com/slient2009/DistanceMeasurement

1. Getting Started

We conducted all the experiments on the server with `Ubuntu 20.04` equipped with 26 Intel(R) Xeon(R) Gold 6230R CPU cores with 2.10 GHz and 220 GB of memory.
To reproduce the experimental result in our paper, a server with same or better performance is recommended.

You can run AFLGo Building Script to do everything for you instead of manually go through **Step 1.1**.
Be careful in these steps we would download, build and install LLVM 11.0.0 from source, which may have unexpected impacts on compiler toolchain in current system.

1.1 Environment

1. Install LLVM **11.0.0** with Gold-plugin. Then make sure that the following commands successfully executed:

```
# Install LLVMgold into bfd-plugins
mkdir /usr/lib/bfd-plugins
cp /usr/local/lib/libLTO.so /usr/lib/bfd-plugins
cp /usr/local/lib/LLVMgold.so /usr/lib/bfd-plugins
```

2. Install other prerequisite

```
sudo apt-get update
sudo apt-get install python3
sudo apt-get install python3-dev
sudo apt-get install python3-pip
sudo apt-get install pkg-config
sudo apt-get install autoconf
sudo apt-get install automake
sudo apt-get install libtool-bin
sudo apt-get install gawk
sudo apt-get install libboost-all-dev  # boost is not required if you use
gen_distance_orig.sh in step 7
python3 -m pip install networkx==2.8.8 # May vary by different python
versions, see the case statement in build.sh
python3 -m pip install pydot
python3 -m pip install pydotplus
```

3. Compile AFLGo-variants fuzzer, LLVM-instrumentation pass and the distance calculator

3. Compile AFLGo-variants fuzzer, LLVM instrumentation pass and the distance calculator

```
export CXX=`which clang++`
export CC=`which clang`
export LLVM_CONFIG=`which llvm-config`

git clone https://github.com/slient2009/DistanceMeasurement.git

export AFLGO_VARIANTS='/path/to/aflgo-arithmetic-appr'

cd AFLGO_VARIANTS/scripts
./compile.sh        # compile LLVM pass and distance calculator in one run
```

1.2 Configuration

To run aflgo fuzzer, some system settings are required before a successful run.

Just follow the instructions of aflgo when starting fuzzing, for example:

```
[-] Hmm, your system is configured to send core dump notifications to an
    external utility. This will cause issues: there will be an extended delay
    between stumbling upon a crash and having this information relayed to the
    fuzzer via the standard waitpid() API.

    To avoid having crashes misinterpreted as timeouts, please log in as root
    and temporarily modify /proc/sys/kernel/core_pattern, like so:

    echo core >/proc/sys/kernel/core_pattern
[*] Checking CPU scaling governor...

[-] Whoops, your system uses on-demand CPU frequency scaling, adjusted
    between 976 and 3906 MHz. Unfortunately, the scaling algorithm in the
    kernel is imperfect and can miss the short-lived processes spawned by
    afl-fuzz. To keep things moving, run these commands as root:

    cd /sys/devices/system/cpu
    echo performance | tee cpu*/cpufreq/scaling_governor

    You can later go back to the original state by replacing 'performance' with
    'ondemand'. If you don't want to change the settings, set AFL_SKIP_CPUFREQ
    to make afl-fuzz skip this check - but expect some performance drop.
```

1.3 Code Tree

The artifact contains `9 AFLGo-variants code`, `6 CVEs` as benchmark, and `scripts` which automatically analysis PoC lineage and mutation effectiveness and draw corresponding figures.

```
$ git clone https://github.com/slient2009/DistanceMeasurement.git
$ cd DistanceMeasurement
$ tree -L 3
.
├── aflgo-variants
|   ├── aflgo-arithmetic-appr       # AFLGo-variants dir, same as AFLGo, with
different in distance calculation code
```

```
|   ├── aflgo-arithmetic-bblk
|   ├── aflgo-arithmetic-func
|   ├── aflgo-harmonic-appr
|   ├── aflgo-harmonic-bblk
|   ├── aflgo-harmonic-func
|   ├── aflgo-shortest-appr
|   ├── aflgo-shortest-bblk
|   └── aflgo-shortest-func
├── analysis
|   ├── mutation-assess
|   |   ├── cve-2016-4487-arithmetic-appr-run-6-decrease-cactusplot-ylim.png
|   |   ├── cve-2016-4487-arithmetic-appr-run-6-decrease-cactusplot-ylim.png.eps
|   |   ├── cve-2016-4487-arithmetic-appr-run-6-scatter-decrease-by-time.png
|   |   ├── cve-2016-4487-arithmetic-appr-run-6-scatter-decrease-by-time.png.eps
|   |   ├── ...
|   ├── mutation-assess.py          # mutation assessment script, output figures
stored in mutation-assess dir
|   ├── PoC-lineage-analysis
|   |   ├── PoC-lineage-analysis-with-dis-cve-2016-4487-9-new.png
|   |   ├── PoC-lineage-analysis-with-dis-cve-2016-4487-9-new.png.eps
|   |   ├── PoC-lineage-analysis-with-dis-cve-2016-4489-0-new.png
|   |   ├── PoC-lineage-analysis-with-dis-cve-2016-4489-0-new.png.eps
|   |   ├── ...
|   ├── PoC_lineage_analysis.py     # PoC lineage analysis script, output figures
stored in PoC-lineage-analysis dir
|   └── PoC_lineage_length_hist.py
├── benchmark
|   ├── CVE-2016-4487
|   |   ├── CVE-2016-4487.patch     # patch code of CVE
|   |   ├── cxxfilt-patched         # patched program binary of cxxfilt
|   |   ├── cxxfilt-unpatch         # unpatch program binary of cxxfilt
|   |   ├── setup-CVE-2016-4487.sh  # download binutils-cxxfilt, compile, and
start fuzzing on cxxfilt
|   |   ├── setup-cxxfilt-CVE-2016-4487-patched.sh  # compile the unpatch program
binary of cxxfilt
|   |   ├── setup-cxxfilt-CVE-2016-4487-unpatch.sh  # patch the CVE-2016-4487,
compile the patched program binary of cxxfilt
|   |   ├── triage-crash-cve-2016-4487-cxxfilt.sh   # triage crashes to determine
the PoC of CVE-2016-4487
|   |   └── distance_log            # log file in fuzzing
|   ├── CVE-2016-4489
|   ├── CVE-2016-4490
|   ├── CVE-2016-4492
|   ├── CVE-2017-9047
|   └── CVE-2018-8807
├── README.md
```

The differences among the 9 AFLGo-variants mainly locate in the code related to distance calculation. (./llvm_mode/afl-llvm-pass.so.cc, ./afl-fuzz.c, ./scripts/distance.py, ./scripts/gen_distance_fast.py)

2. Apply AFLGo-variants on target program (cxxfilt: CVE-2016-4490)

2.1 Start Fuzzing

1. You need to compile each AFLGo-variant respectively. For `aflgo-arithmetic-appr`, you need

```
#!!! USE THE PATH TO ARTIFACT
cd /path/to/artifact/DistanceMeasurement/aflgo-variants/aflgo-arithmetic-appr/script
./compile.sh
```

2. Before starting fuzzing, the `WORKDIR` variable shall be exported determine where the artifact is and the `AFLGo variant` shall be configured in the setup-CVE-XXXX-YYYY.sh

```
git clone git://sourceware.org/git/binutils-gdb.git cxxfilt-CVE-2016-4492
cd cxxfilt-CVE-2016-4492
git checkout 2c49145

mkdir obj-aflgo; mkdir obj-aflgo/temp

export WORKDIR=/home/wenake/tmp/zenodo/DistanceMeasurement      # define workdir as where the DistanceMeasurement is
export AFLGO=$WORKDIR/aflgo-variants/aflgo-harmonic-appr        # switch to other aflgo variants here

export SUBJECT=$PWD; export TMP_DIR=$PWD/obj-aflgo/temp
export CC=$AFLGO/afl-clang-fast; export CXX=$AFLGO/afl-clang-fast++
```

3. Switch to the directory of target CVE (e.g. CVE-2016-4490) and run setup-CVE-2016-4490.sh

```
cd $WORKDIR/benchmark/CVE-2016-4490/
./setup-CVE-2016-4490 10  #run 10 parallel processes
```

The timeout is set to 24h and the exploit time is set to 21h as default aflgo setting.

However, the fuzzing task of CVE-2016-4490 is expected to finish in half hour.

2.2 Code Tree

```
$ tree ./benchmark/CVE-2016-4490/
.
├── obj-aflgo
    ...
    ├── temp                # aflgo preprocessing dir
        ├── BBcalls.txt
        ├── BBnames.txt
        ├── BBtargets.txt
        ├── callgraph.distance.txt
        ├── distance.cfg.txt
        ├── dot-files
```

```
        ├── Fnames.txt
        ├── Ftargets.txt
        └── step0.log
├── obj-dist
    ...
    ├── out
        ├── out_0           # aflgo fuzzing dir
            ├── crashes
            ├── distance_log
            ├── fuzz_bitmap
            ├── fuzzer_stats
            ├── hangs
            ├── plot_data
            └── queue
        ├── out_1
        ├── out_2
        ...
```

2.3 Triage

There are many crashes detected in fuzzing, but only few of them are related to the target vulnerability.

Here we compile the unpathed target program and patched target program.
We determine whether a testcase exposes the target vulnerability by executing the failing inputs on the patched version of the target program.
If a failing testcase passes on the patched version, it is said to witness the target vulnerability.

The unpatched program and patched program are provided as well as the patch code.
The patch code is determined by carefully analysis the patch commit on the codebase of cxxfilt.

```
./benchmark/CVE-2016-4490/cxxfilt-patched
./benchmark/CVE-2016-4490/cxxfilt-unpatch
./benchmark/CVE-2016-4490/CVE-2016-4490.patch
```

cve and patch links:

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-4490
https://gcc.gnu.org/bugzilla/show_bug.cgi?id=70498
https://gcc.gnu.org/git/?p=gcc.git&a=commit;h=9e6edb946c0e9a2c530fbae3eeace148eca0de33

Triage script

An automatic triage script is provided.
The patched program, unpatched program, and the crash directory of fuzzer shall be assigned to script.

Moreover, we could triage out 1 or many PoC from all crashes generated by setting the last parameter.
(Generally, the first PoC is all we need, so just set the last parameter to 1)

```
#!!!! WORKDIR variable is needed !!!!
export WORKDIR=/home/wenake/tmp/zenodo/DistanceMeasurement
$WORKDIR/benchmark/CVE-2016-4490/triage-crash-cve-2016-4490-cxxfilt.sh
$WORKDIR/benchmark/CVE-2016-4490/cxxfilt-patched $WORKDIR/benchmark/CVE-2016-
4490/cxxfilt-unpatch $WORKDIR/benchmark/CVE-2016-4490/cxxfilt-CVE-2016-4490/obj-
dist/out/out_0/crashes 1
```

The info of the first PoC (if it exists) will be written into ./benchmark/CVE-2016-4490/cxxfilt-CVE-2016-4490/obj-dist/out/out_0/crashes/real_crash.txt, like:

```
id:000017,12006250,sig:11,src:000311,op:havoc,rep:2,dis:1875.13
```

Following the naming convention of AFL, this PoC is the `17th` crash of all crashes, derived from `seed 311` after `havoc and replacing` operation.

Most importantly, it is generated at `12006250ms` after the fuzzer started and that is the TTE (Time-To-Exposure) of the fuzzing process.

3. Data Analysis (cxxfilt: CVE-2016-4490)

3.1 Overall Comparison among Variants

Now we could compare the performance between different distance calculation methods and granularities.

By gathering all the TTE of all fuzzing process of different AFLGO-Variants, we could list the `Table.II` and `Table.III` in our paper:

TABLE II: Performance Comparison among Different Distance Calculation Methods.

CVE-ID	Method	Runs	μTTE	Factor	$\hat{A}_{12}$
2016-4487	harmonic	10	291	-	-
	arithmetic	10	**114.26**	2.55	0.56
	closest	10	142.52	2.04	0.56
2016-4489	harmonic	9	258.51	-	-
	arithmetic	10	**139.72**	1.85	0.75
	closest	10	146.59	1.76	0.54
2016-4490	harmonic	10	**32.62**	-	-
	arithmetic	10	34.26	0.95	0.49
	closest	10	44.47	0.73	0.30
2016-4492	harmonic	9	451.62	-	-
	arithmetic	8	**390.66**	1.16	0.64
	closest	9	489.39	0.92	0.44
2018-8807	harmonic	9	1002.95	-	-
	arithmetic	9	**869.02**	1.15	0.64
	closest	8	1069.13	0.94	0.49
2017-9047	harmonic	10	223.08	-	-
	arithmetic	10	162.33	1.37	0.58
	closest	10	**97.45**	2.29	0.68
			Mean Factor		Mean $\hat{A}_{12}$
	arithmetic		1.51		0.61
	closest		1.45		0.50

[1] The distance graunlarity is set to approximated basic-block-level (appr) distance.
[2] The statistically significant values of $\hat{A}_{12}$ and the smallest TTE is highlighted in bold.
[3] A run that does not reproduce the vulnerability within 24 hours receives a TTE of 24 hours.

TABLE III: Performance Comparison among Different Distance Granularities.

CVE-ID	Granularity	Runs	μTTE	Factor	$\hat{A}_{12}$
2016-4487	func	10	126.81	-	-
	appr	10	**114.26**	1.11	0.51
	bblk	10	148.02	0.86	0.61
2016-4489	func	10	219.64	-	-
	appr	10	**139.72**	1.57	0.81
	bblk	9	241.59	0.91	0.56
2016-4490	func	10	**32.15**	-	-
	appr	10	34.26	0.94	0.45
	bblk	10	35.36	0.91	0.43
2016-4492	func	9	376.36	-	-
	appr	8	390.66	0.96	0.66
	bblk	10	**268.32**	1.40	0.53
2018-8807	func	10	**837.95**	-	-
	appr	9	869.02	0.96	0.50
	bblk	9	953.46	0.88	0.40
2017-9047	func	10	111.19	-	-
	appr	10	162.33	0.68	0.44
	bblk	10	**80.21**	1.39	0.56
			Mean Factor		Mean $\hat{A}_{12}$
	appr		1.04		0.56
	bblk		1.06		0.52

[1] The distance calculation method is set to the arithmetic mean.
[2] The statistically significant values of $\hat{A}_{12}$ and the smallest TTE is highlighted in bold.
[3] A run that does not reproduce the vulnerability within 24 hours receives a TTE of 24 hours.
[4] func, appr, bblk is short for function-level distance, approximated basic-block-level distance, and basic-block-level distance respectively.

3.2 PoC lineage analysis

We add some log code to record the details of seed, testcase, and crash:

```
// 1 for every testcase
// 2 for add_to_queue testcase
// 3 for save_if_interesting to write a `queue/id:the_id,xxxx` file
// 4 for crashes to write a `crashes/id:the_id,xxxx` file
static u32 last_distance_log_seq=0;
static u64 last_distance_log_time=0;
static void log_distance(double d, int type, int the_id){
  if(type==2){
    fprintf(distance_log_file,
"type:%d,mindis:%.2f,dis:%.2f,maxdis:%.2f,totexec:%llu,time:%llu\n", type,
min_distance, d, max_distance, total_execs, get_cur_time()-start_time);
    fflush(distance_log_file);
  }
  else if(type==3){
    fprintf(distance_log_file,
"type:%d,mindis:%.2f,dis:%.2f,maxdis:%.2f,totexec:%llu,time:%llu,queueid:%d\n",
type, min_distance, d, max_distance, total_execs, get_cur_time()-start_time,
the_id);
    fflush(distance_log_file);
  }
  else if(type==4){
    fprintf(distance_log_file,
"type:%d,mindis:%.2f,dis:%.2f,maxdis:%.2f,totexec:%llu,time:%llu,crashid:%d\n",
type, min_distance, d, max_distance, total_execs, get_cur_time()-start_time,
the_id);
    fflush(distance_log_file);
  }
  else{
    u64 t = get_cur_time();
    if(t - last_distance_log_time >= LOG_DISTANCE_TIME_GAP){
      fprintf(distance_log_file,
"type:%d,mindis:%.2f,dis:%.2f,maxdis:%.2f,totexec:%llu,time:%llu,src:%d\n", type,
min_distance, d, max_distance, total_execs, t-start_time, current_entry);
      fflush(distance_log_file);
      last_distance_log_time = t;
    }
  }
}
```

The output file is `./benchmark/CVE-2016-4490/cxxfilt-CVE-2016-4490/obj-dist/out/out_0/distance_log`:

The details information includes the distance, generated time, father seed id of current testcase, seed and crash. Moreover, the global max and min distance of all seeds are recorded.

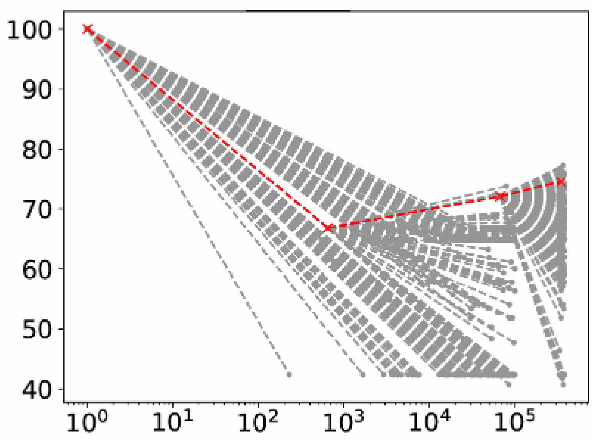
```
# type | min_distance cur_distance max_distance | total_execs | offtime | id:xxx
...
type:2,mindis:670.11,dis:2471.54,maxdis:3388.42,totexec:360,time:81
type:3,mindis:670.11,dis:2369.78,maxdis:3388.42,totexec:441,time:101,queueid:22
type:2,mindis:670.11,dis:2369.78,maxdis:3388.42,totexec:441,time:101
type:1,mindis:670.11,dis:3041.42,maxdis:3388.42,totexec:450,time:104,src:0
type:3,mindis:670.11,dis:2519.03,maxdis:3388.42,totexec:483,time:111,queueid:23
type:2,mindis:670.11,dis:2519.03,maxdis:3388.42,totexec:483,time:111
type:3,mindis:670.11,dis:2546.53,maxdis:3388.42,totexec:498,time:115,queueid:24
type:2,mindis:670.11,dis:2546.53,maxdis:3388.42,totexec:498,time:115
type:3,mindis:670.11,dis:2415.55,maxdis:3388.42,totexec:528,time:124,queueid:25
...
```

Now we could run the `PoC_lineage_analysis.py` to draw the lineage of PoC.

You need to set the work directory `fuzz_out_dir` of a fuzzing process, for example `./benchmark/CVE-2016-4490/cve-2016-4490-cxxfilt-arithmetic-appr/obj-dist/out/out_0/`.

```
# !!! you need manully set fuzz_out_dir variable inside the python script !!!
fuzz_out_dir = '/mnt/data/wenake/aflgo-formal-experiments/cve-2016-4490-cxxfilt/cve-2016-4490-cxxfilt-arithmetic-appr/obj-dist/out/out_'+str(i)+'/'
```

The script will automatically read the first PoC among all crashes, analysis the information in `distance_log` file. Then, the lineage of PoC will be generated:



CVE-2016-4490

Also, the `Lineage Length Distribution Histogram` is produced by running `./analysis/PoC_lineage_length_hist.py`.

You can use the lineage length data to produce the histogram of your own experiments.

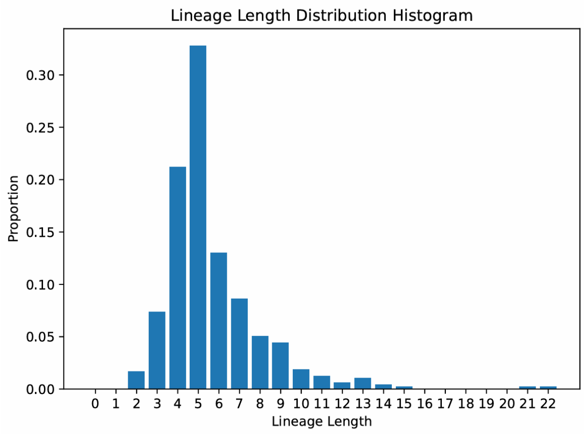


Fig. 3: The Distribution of Lineage Length of PoC.

3.3 Mutation Assessment

Run `mutation-assess.py` to assess the effectiveness of mutation and draw figures.

You need set the `queue_dir` in script, for example `./benchmark/CVE-2016-4490/cve-2016-4490-cxxfilt-arithmetic-appr/obj-dist/out/out_0/queue/`.

```
# !!! you need manully set queue_dir variable inside the python script !!!
queue_dir = '/mnt/data/wenake/aflgo-formal-experiments/cve-2016-4490-cxxfilt/cve-2016-4490-cxxfilt-arithmetic-appr/obj-dist/out/out_'+str(i)+'/queue/'
```

then run the python script and get the statistics report as follow:

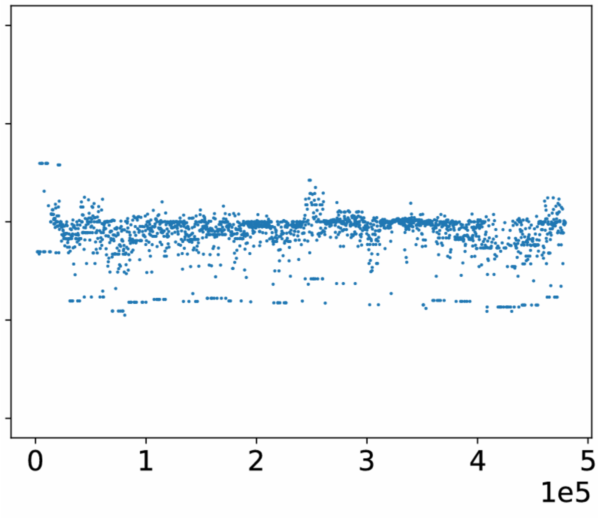
```
$ python mutation-assess.py
cve-2016-4490-arithmetic-appr-run-7
MAXN_CREASH_TIME: 479596.8
seed number: 861 testcase number: 4705 MAXN_CREASH_TIME 479596.8
cve-2016-4490-arithmetic-appr-run-7
Distance Decrease:
 seed number: 3978
 min = -0.48, mean = -0.06, median = -0.03, max = 0.42 # `Decrease` Statistics
 Significat = 0.00 Trival = 0.71 Positive = 0.29
```

We perform a quantitative analysis of the capability of the existing mutation strategy to generate high-quality testcases. Specifically, we assess the ability of mutation strategy to generate closer testcases by `Decrease`, which is defined as:

$$Decrease = \frac{Distance(T) - Distance(S)}{Distance(S)}$$

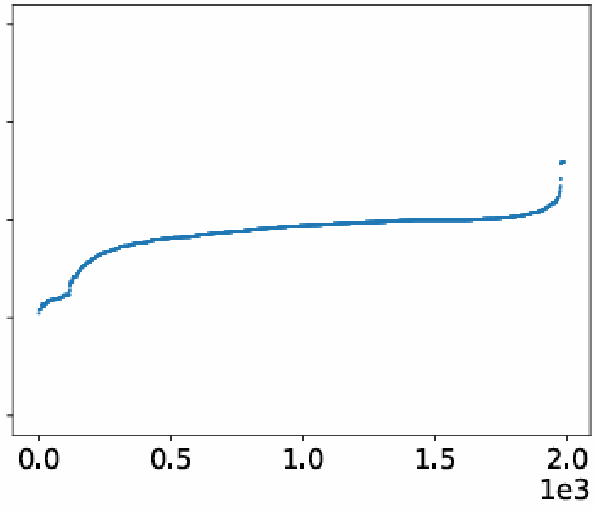
where the testcase T is directly derived from seed S.

If the $Decrease$ exhibits an extremely small negative value, it implies that the distance of the descendant testcase is significantly reduced relative to its parent seed, thereby increasing the likelihood of approaching and triggering the target vulnerability.
Conversely, a large positive value of $Decrease$ indicates that the descendant testcases have significantly greater distances compared to their parent seeds, which is an unfavorable outcome.
In the scenario where $Decrease$ is close to 0, it is interpreted as only a negligible distance difference between the descendant testcase and its parent seed.



CVE-2016-4490

Furthermore, we have drawn the cactus plot of distance $Decrease$ as figure above.
Although the distance of most testcases decreased compared to their parent seeds, the decrease proportion is negligible.



CVE-2016-4490