USENIX21 Artifact Evaluation Instructions

1 Artifacts

Our SelectiveTaint tool consists of two parts: the static analysis part and the static rewriting part. As static rewriting is built for 32-bit for legacy applications comparison, it is built in Ubuntu 14.04 32 bit OS. As 14.04 does not support new versions of angr, our static analysis part is built in Ubuntu 20.04 64 bit OS.

1.1 Download VMs

The very first step is to download the virtual machines hosting these two parts.

As for static analysis, download from this google drive link:

`https://drive.google.com/file/d/12tOhGcsr1JrAkHNSlB3m7vdeODNOrpsU/view?usp=sharing`

As for static rewriting, download from this google drive link:

`https://drive.google.com/file/d/11epjWVhdP_utp-2dm2EpUVvyIQZSikSP/view?usp=sharing`

1.2 Load VMs

Load the VM application in virtualbox. Make sure the host machine has at least 32GB memory as the VM may require certain amount of memory.

The VM administrator user:
user name: sec
password: sec

Specific instructions for 14.04 VM Press and hold SHIFT key when you start the VM and see the logo. Sometimes, you may need to restart or shut down several times, until you see the kernel selection at the beginning. Make sure to select kernel version 3.13, otherwise Intel Pin and others will not work properly: Go to Advanced options for Ubuntu: select 3.13 kernel.

2 Testing Static Analysis

Static analysis code is in VM USENIX-artifact-2.ova. The applications we are going to test are 29 applications, including 16 Unix utilities , 5 network daemons, and nine other applications:

As we do not have SPEC 2017 (it is a proprietary software) as suggested by the reviewers, we cannot evaluate SelectiveTaint on it. Therefore, we collect two kinds of open source programs as the benchmarks (so that others can also easily evaluate), covering both file I/O and network I/O. These 16 Unix utilities are: tar (version 1.27.1), gzip (version 1.3.13), bzip2 (version 1.0.3), scp (version 3.8), cat, comm, cut, grep, head, nl, od, ptx, shred, tail, truncate, uniq (version 8.21) and grep (version 2.16), and 5 network daemons are: email server exim (version 4.80), general-purpose distributed memory caching system Memcached (version 1.4.20), FTP server ProFTPD (version 1.3.5), web server lighttpd (version 1.4.35) and nginx (version 1.4.0).

Please change your directory to ∼/artifact/selectivetaint_static_analysis. Please read README.md as the commands are inside. Please execute the following commands one by one. Be aware some commands may execute more than 2 days, depending on the machine computing capability.

Fundamentally, you need to install python3, ipython, angr, etc. As in our environment, they are already installed.

2.1 Newly added progams

The commands to test newly added programs are:

python3 static.py -input ./tar -taintsources read fscanf
python3 static.py -input ./gzip -taintsources read _IO_getc
python3 static.py -input ./bzip2 -taintsources fread fgetc
python3 static.py -input ./scp -taintsources read
python3 static.py -input ./cat -taintsources read fscanf
python3 static.py -input ./comm -taintsources fscanf
python3 static.py -input ./cut -taintsources fgetc fscanf __fread_chk
python3 static.py -input ./grep -taintsources read fscanf fread_unlocked
python3 static.py -input ./head -taintsources read fscanf
python3 static.py -input ./nl -taintsources fscanf
python3 static.py -input ./od -taintsources fgetc fscanf fread_unlocked __fread_unlocked_chk
python3 static.py -input ./ptx -taintsources fread fscanf
python3 static.py -input ./shred -taintsources fscanf __read_chk fread_unlocked
python3 static.py -input ./tail -taintsources read fscanf
python3 static.py -input ./truncate -taintsources fscanf
python3 static.py -input ./uniq -taintsources fscanf
python3 static.py -input ./exim -taintsources fgetc fread fscanf __IO_getc recv
python3 static.py -input ./memcached -taintsources read fgets recvfrom
python3 static.py -input ./lighttpd -taintsources read fread
python3 static.py -input ./proftpd -taintsources read fgets __read_chk
python3 static.py -input ./nginx -taintsources read pread64 readv recv

2.2 Unchanged progams

The static analysis for vulnerable programs remains unchanged as the following:

python3 static.py -input ./sox -taintsources read fread fgets _IO_getc __isoc99_scanf __isoc99_fscanf

python3 static.py -input ./tt++ -taintsources read fread fgets _IO_getc gnutls_record_recv fgetc

python3 static.py -input ./dcraw -taintsources fread fscanf __fread_chk _IO_getc fgets jpeg_read_header

python3 static.py -input ./gif2tga -taintsources fread _IO_getc

python3 static.py -input ./gravity -taintsources read getline

python3 static.py -input ./mp3gain -taintsources fread _IO_getc

python3 static.py -input ./nasm -taintsources fread fgets fgetc

python3 static.py -input ./jhead -taintsources fread fgetc

2.3 Sample output

You may redirect the output to a file or output in terminal. A sample output is shown in Figure 1:

"Table 2 Column 3: # Func.: 53", this is Table 2 Column 3 in our paper.

"Table 2 Column 4: # Inst.: 2628", this is Table 2 Column 4.

"init edges: 1046", this is Table 3 Column 2.

"updated edges: 1113", this is Table 3 Column 3.

"time: 2.156066417694092, this is Table 3 Column 7.

"abstract loction count: 657", this is Table 3 Column 4.

"uninitialized var 1 count: 6", this is Table 3 Column 5.

"uninitialized var 2 count: 17", this is Table 3 Column 6.

"uninitialized var 3 count: 48", this is Table 3 Column 7.

"uninitialized var total count: 71", this is Table 3 Column 8.

"1st iteration untainted var count: 600", this is Table 3 Column 9.

"last iteration untainted var count: 600", this is Table 3 Column 10.

"intra procedural analysis iteration times: 110", this is Table 3 Column 11.

"inter procedural analysis iteration times: 2", this is Table 3 Column 12.

3 Testing Static Rewriting

3.1 Set up environment

Static rewriting code is in VM USENIX-artifact.ova. Make sure you follow the instructions above to enter kernel 3.13. The applications we are going to test are 21 applications. Please cd to ∼/artifact/selectivetaint directory. And then set up environment by:

. ./scripts/set_environment.sh

3.2 Instrumentation examples

To use selectivetaint, cd to ∼/artifact/selectivetaint directory.

make clean && make

To instrument all taint logic, for instance, mcf:

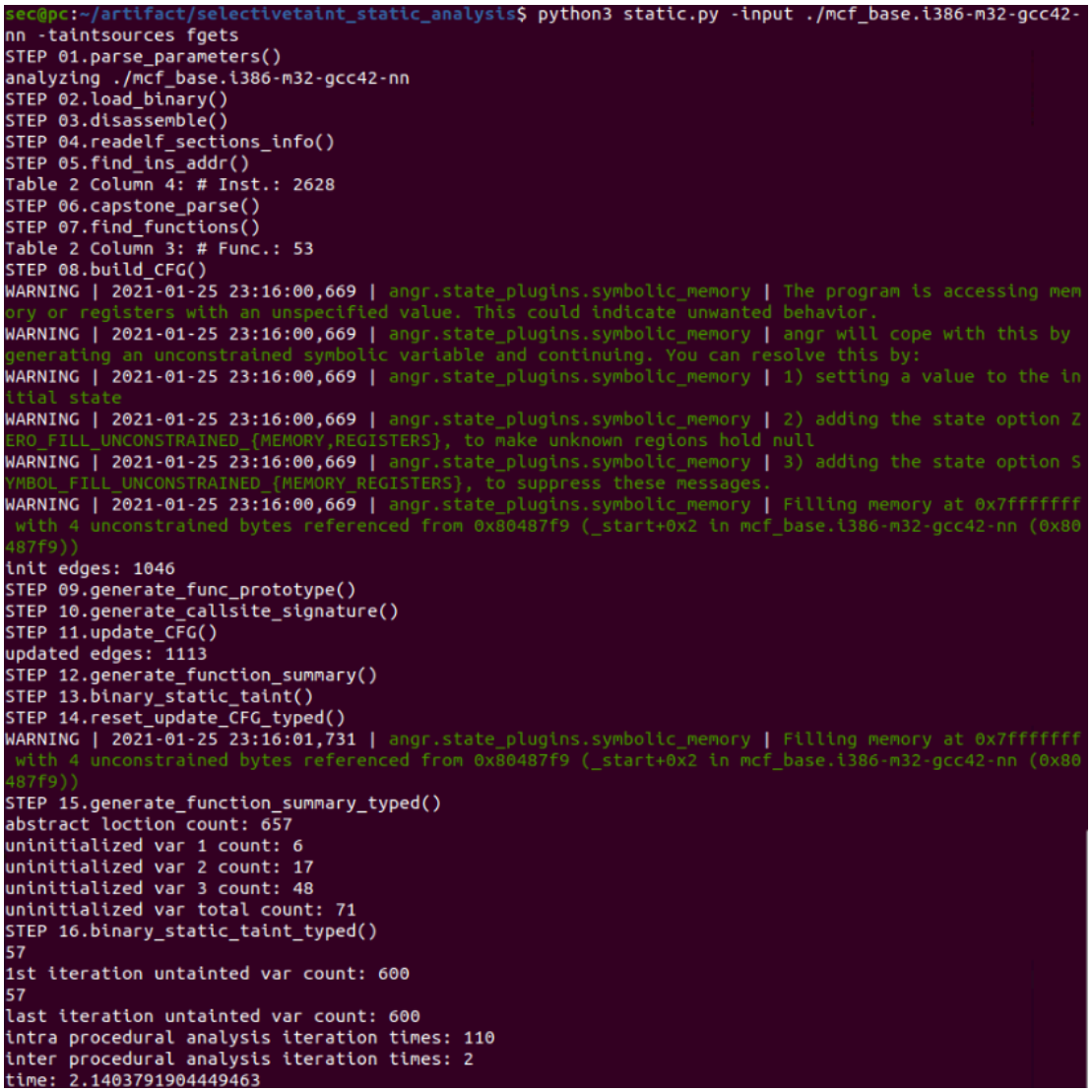

Figure 1: **Static Analysis Sample Output**

./selectivetaint -i ./mcf_base.i386-m32-gcc42-nn -o ./mcf_base.i386-m32-gcc42-nn_all

To selectively instrument selected taint logic, for instance, mcf:

./selectivetaint -i ./mcf_base.i386-m32-gcc42-nn -o ./mcf_base.i386-m32-gcc42-nn_selective -t ./tests/mcf_base.i386-m32-gcc42-nn_tainted_insn_typed_output_file

./tests/mcf_base.i386-m32-gcc42-nn_tainted_insn_typed_output_file is the tainted instructions generated from our static analysis.

3.3 Test table 4. vulnerable CVEs (this part remain unchanged)

We are going to test the following:

sox
tt++
dcraw
ngiflib
gravity
mp3gain
nasm
jhead
nginx

Their corresponding taintall binary and selectivetaint binary are with _all and _selective at the end of the binary name.

For instance:
sox is original binary
sox_all is taintall binary
sox_selective is selectivetaint binary
cd tests and we will do tests here.

3.3.1 sox

For sox, to test taintall,

./sox_all –single-threaded ./poc-CVE-2019-8356.mp3 -t aiff /dev/null channels 1 rate 16k fade 3 norm

Results will be in result_output. Check the content and the example output is in each result backup folder, for instance, for sox, the result will be in 01_sox_tests.

To test selectivetaint,

./sox_selective –single-threaded ./poc-CVE-2019-8356.mp3 -t aiff /dev/null channels 1 rate 16k fade 3 norm

3.3.2 tt++

tt++ is a bit different, you need two terminals
first open a new terminal in this tests directory, type:
python ./tt_server.py
wait until you see:
Waiting for connections on 0.0.0.0:4000
then in the original terminal,
then, execute taintall binary for tt++: ./tt++_all
finally, wait util this line occur:
##########################
T I N T I N + + 2.01.6
# #

The Kickin Tickin DikuMUD Client
# #
Code by Peter Unold, Bill Reis, and Igor van den Hoven
##########################
type enter key to start, wait until we can see:
#NO SESSION ACTIVE. USE: #session {name} {port} TO START ONE.
type this following, press the enter key and wait patiently:
#session test localhost 4000
To test selectivetaint binary:
Repeat the above steps but for tt++_selective

3.3.3 dcraw

./dcraw_all ./poc-CVE-2018-19655
./dcraw_selective ./poc-CVE-2018-19655

3.3.4 gif2tga

./gif2tga_all ./stack-buffer-overflow-ngiflib-c-543.poc
./gif2tga_selective ./stack-buffer-overflow-ngiflib-c-543.poc

3.3.5 gravity

./gravity_all ./test.gravity
./gravity_selective ./test.gravity

3.3.6 mp3gain

./mp3gain_all -f ./00348-aacgain-stackoverflow-copy_mp
./mp3gain_selective -f ./00348-aacgain-stackoverflow-copy_mp

3.3.7 nasm

./nasm_all -f elf ./poc-CVE-2019-8343.asm
./nasm_selective -f elf ./poc-CVE-2019-8343.asm

3.3.8 jhead

./jhead_all ./poc-CVE-2018-6612
./jhead_selective ./poc-CVE-2018-6612

3.3.9 nginx

As for nginx, may need to restart vm several times, be patient.
Run the following manually several times (e.g., 3 times)
sudo LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. ./nginx_all -s stop -c /usr/local/nginx/nginx.conf

sudo LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. ./nginx_all -c /usr/local/nginx/nginx.conf

It's okay if the first command returns some error message.

Run this in another terminal:

python nginx_rop_plain_try.py

See output in result_output

Rerun the above steps for ngin_selective

3.4 Test Figure 7.

For the artifact evaluation since VM performance results largely depends on the host machine and are different from our physical machine, we therefore do __not__ expect artifact reviewers to generate the raw data themselves. We instead provide the detailed instructions of how to reproduce Figure 7.

Please replace the all instrumented binary with libdft and selective taint version for the 10 benchmarks and rerun the tests for libdft and SelectiveTaint. Example:

3.4.1 Test taintall example command

./mcf_base.i386-m32-gcc42-nn_all inp.in > /dev/null

3.4.2 Test libdft example command

/home/sec/artifact/pin-2.14/pin -t /home/sec/artifact/selectivetaint/tests/libdft/-tools/libdft.so – ./mcf_base.i386-m32-gcc42-nn inp.in > /dev/null

When prompted with the following error message:

E: Attach to pid 2509 failed.

E: The Operating System configuration prevents Pin from using the default (parent) injection mode.

E: To resolve this, either execute the following (as root):

E: $ echo 0 > /proc/sys/kernel/yama/ptrace_scope

E: Or use the "-injection child" option.

E: For more information, regarding child injection, see Injection section in the Pin User Manual.

E: Killed

Use sudo su to switch to sudoer and type:

$ echo 0 > /proc/sys/kernel/yama/ptrace_scope

And then enter "exit" to switch to normal user and execute the libdft command again.

3.4.3 Test selectivetaint example command

./mcf_base.i386-m32-gcc42-nn_selective inp.in > /dev/null

3.4.4 Data in Figure 7.

Instructions for 16 Unix utilities used in evaluation:

./tar -cf a.tar ./dir
./gzip -cf ./1KBFILE
./bzip2 -zkf 1KBFILE
sshpass -p "AAaa1234" ./scp usenix@164.107.119.58:~/OVERVIEW .
./cat textfile
./comm textfile textfile1
./cut -b 1,2,3 textfile
./grep "count" textfile
./head textfile
./nl textfile
./od -c ./textfile
./ptx textfile
./shred ./textfile
./tail textfile
./truncate -s 100 textfile
./uniq textfile

3.4.5 Test exim

Please cd to /home/sec/artifact/selectivetaint/exim_dir

./exim -bt sec

3.4.6 Test memcached

Please set up memcached by:

./memcached -d -p 11211 -u nobody -c 1024 -m 64

We already store data in memcached (which reviewer certainly does not need to perform, they are here only for your information):

telnet localhost 11211
set foo 0 0 3
To get data from database:
echo "get foo" — ncat localhost 11211
After testing, if tester would like to release the port, one way is:
sudo su
ps aux — grep memcached
kill -9 PID
exit (exit sudoer)

3.4.7 Test proftpd

Please cd to /home/sec/artifact/selectivetaint/proftpd_dir

Set up the ftp server:
./proftpd -c /etc/proftpd/proftpd.conf
For instrumented binary, if not found the library, use the following:

sudo LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. ./proftpd_all -c /etc/proftpd/proftpd.conf

Command to test:

time wget ftp://localhost/1KBFILE

3.4.8 Test lighttpd

Please cd to /home/sec/artifact/selectivetaint/lighttpd_dir

Set up the http server:

./lighttpd -f /etc/lighttpd/lighttpd.conf

For instrumented binary, if not found the library, use the following:

sudo LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. ./lighttpd_all -f /etc/lighttpd/lighttpd.conf

Command to test:

time wget ftp://localhost/1KBFILE

3.4.9 Test nginx

The test mostly remain unchanged:

Please cd to /home/sec/artifact/selectivetaint/nginx_dir and start nginx:

Run the following manually several times (e.g., 3 times)

sudo LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. ./nginx_all -s stop -c /usr/local/nginx/nginx.conf

sudo LD_LIBRARY_PATH=$LD_LIBRARY_PATH:. ./nginx_all -c /usr/local/nginx/nginx.conf

It's okay if the first command returns some error message.

Repeat this for libdft and SelectiveTaint binary.

The tested 1KB files can be accessed through localhost at 127.0.0.1 when nginx is running.

3.4.10 Data in Figure 7.

The data in Figure 7 is:

	tar	gzip	bzip2	scp	cat	comm	cut	grep
libdft	1.39	3.86	4.89	5.86	3.54	3.87	4.54	4.85
StaticTaintAll	1.10	3.07	3.85	3.58	2.33	2.65	3.06	3.72
SelectiveTaint	1.02	2.56	3.21	2.43	2.01	2.42	2.82	3.41

	head	nl	od	ptx	shred	tail	truncate	uniq	average
libdft	3.81	3.78	4.75	4.59	4.41	3.98	3.77	3.74	4.10
StaticTaintAll	2.20	2.17	3.67	3.13	3.01	2.34	2.02	2.24	2.76
SelectiveTaint	1.91	1.93	2.93	2.73	2.71	2.03	1.82	2.01	2.37

	exim	memcached	proftpd	lighttpd	nginx	average
libdft	3.76	3.38	3.72	2.21	1.38	2.89
StaticTaintAll	2.13	2.04	2.23	1.72	1.25	1.87
SelectiveTaint	1.81	1.72	1.91	1.53	1.12	1.62