Android Security Lab-17

Aakash R
CB.SC.P2CYS23011

Fuzzing

“Fuzzing” technique to test the security and vulnerabilities of crypto libraries in Android applications.

The tools used for fuzzing in this lecture include

1. **OSS-Fuzz:** Continuous Fuzzing Service for Open Source Software
2. **ClusterFuzz:** Scalable Fuzzing Infrastructure
3. **FuzzBench:** Fuzzer benchmarking as a service

which are automated processes used by Google for open-source projects.

Fuzzing involves testing the behaviour and vulnerabilities of functions by manipulating inputs, such as strings or images.

These are all common fuzzing tools, but we will be discussing the CDF (crypto differential fuzzing) tool.

CDF (crypto differential fuzzing)

CDF is a tool to automatically test the correctness and security of cryptographic software. CDF can detect implementation errors, compliance failures, side-channel leaks, and so on.

CDF implements a combination of unit tests with "differential fuzzing", an approach that compares the behaviour of different implementations of the same primitives when fed edge cases and values maximizing the code coverage.

Unlike general-purpose fuzzers and testing software, CDF is:

- **Smart**: CDF knows what kind of algorithm it's testing and adapts to the tested functions.
- **Fast**: CDF tests only what needs to be tested and parallelizes its tests as much as possible.
- **Polyvalent**: CDF isn't specific to any language or API, but supports arbitrary executable programs or scripts.
- **Portable**: CDF will run on any Unix or Windows platform, since it is written in Go without any platform-specific dependencies.

The purpose of CDF is to provide more efficient testing tool to developers and security researchers, being more effective than test vectors and cheaper than manual audit of formal verification.

Installation

Requirements

CDF is coded in Go, the current version has been developed using Go 1.8. It has no dependencies outside of Go's standard library.

However, we provide example programs to be tested using CDF, which are in C, Python, C++, Java and Go and require specific crypto libraries to be run. Currently required libraries are:

- CryptoPP
- OpenSSL
- BouncyCastle
- PyCrypto
- Cryptography.io

git clone https://github.com/kudelskisecurity/cdf.git

```
(venv) [ 17]$ git clone git@github.com:kudelskisecurity/cdf.git
Cloning into 'cdf'...
remote: Enumerating objects: 112, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 112 (delta 0), reused 0 (delta 0), pack-reused 111
Receiving objects: 100% (112/112), 90.26 KiB | 126.00 KiB/s, done.
Resolving deltas: 100% (44/44), done.
(venv) [ 17]$ cd cdf
(venv) [ cdf]$ ls
cdf-lib      config.json.enc   config.json.prf  LICENSE  makefile
config.json  config.json.oaep  examples         main.go  README.md
(venv) [ cdf]$
```

First build the cdf binary.
make

```
(venv) [ 17]$ cd cdf
(venv) [ cdf]$ make
go build -o cdf main.go
(venv) [ cdf]$ ls cdf
cdf
(venv) [ cdf]$
```

make examples-all will build all the examples.

While **make examples-go** will only build the Go examples.

```
(venv) [ cdf]$ make examples-go
cd examples/; make go
make[1]: Entering directory '[redacted]/S3/01: Android Security/classes/assig
nment/17/cdf/examples'
go build dsa_sha256_go.go
go build dsa_sha256_java_wrapper.go
go build ecdsa_p256_sha256_go.go
go build ecdsa_p256_sha256_java_wrapper.go
go build enc_aes128ctr_go-flawed.go
go build enc_aes128ctr_go.go
go build hash_md5_go.go
go build hash_sha256_go.go
go build oaep_rsa2048_go-flawed.go
go build oaep_rsa2048_go.go
go build oaep_rsa2048_java_wrapper.go
go build pkcs_rsa2048_go.go
go build pkcssign_rsa_go.go
go build prf_hmacsha256_go.go
make[1]: Leaving directory '[redacted]/S3/01: Android Security/classes/assign
ment/17/cdf/examples'
(venv) [ cdf]$
```

make test will run unit tests (of CDF).

you may want to view usage info by running **cdf -h**

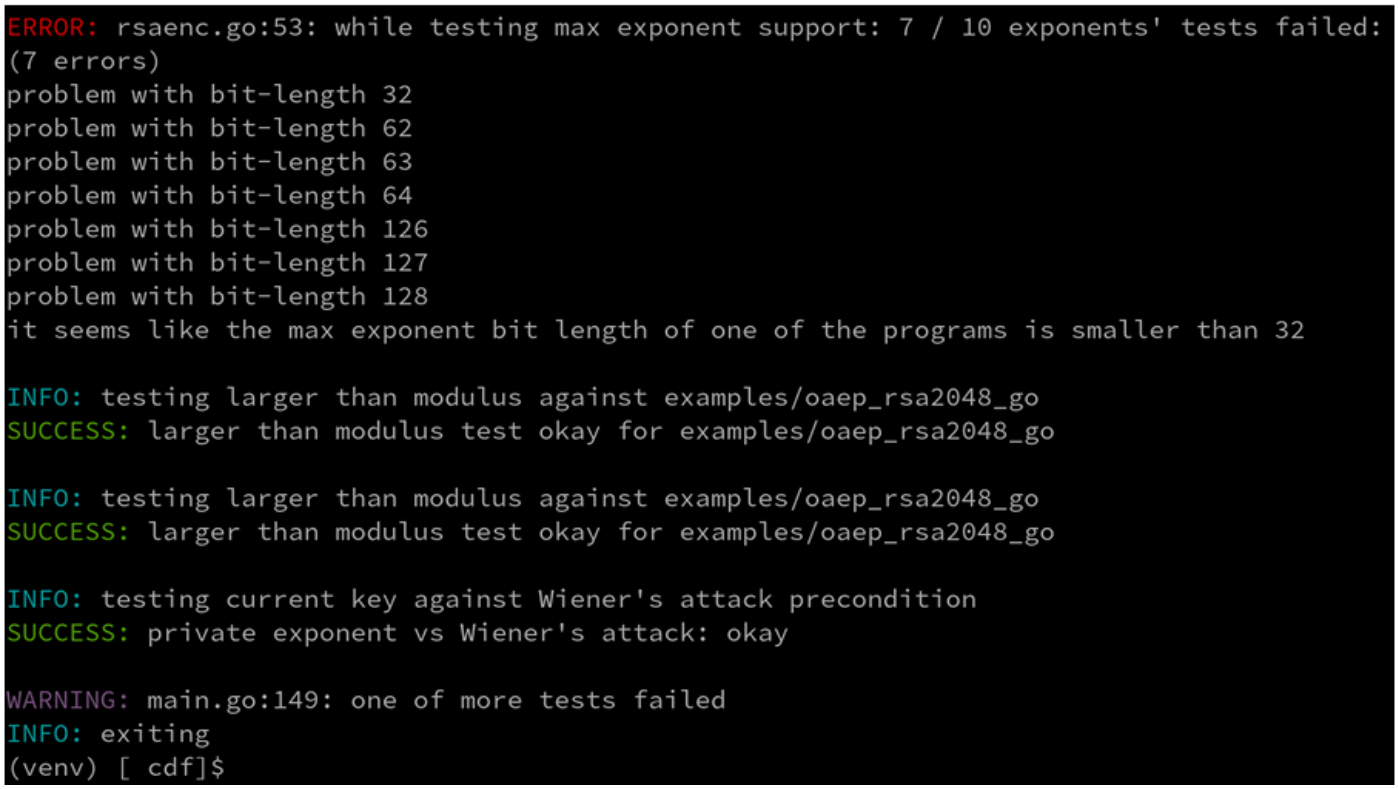
You may then try an example such as the rsaenc interface against the RSA OAEP Go and CryptoPP examples. Viewing CryptoPP as our reference, you can test the Go implementation by doing:

cdf rsaenc /examples/oaep_rsa2048_go /examples/oaep_rsa2048_cryptopp

This command will perform various tests specific to the rsaenc interface.

```
(venv) [ cdf]$ ./cdf rsaenc examples/oaep_rsa2048_go.go examples/oaep_rsa2048_go.go


INFO: Running CDF:
INFO: config: {Seed:5 MinMsgLen:2 MaxMsgLen:214 IncrementMsg:2 MinKeyLen:16 MaxKeyLen:32 Incr
ementKey:8 RsaP:e999d9abbcf3ae2e3261957863bea74f4182cf27e22e4faff461c96ef19a65bf8e85aca934e18
a745e64f7f2be9e150f562dda16e52e0504e4ab53f70c12ebec2ccf4e4c628356c4ebdab3398dfef6274b4c3f5b14
531e4499acd0d59c5da3a03991cf8debb05799d9156ee807c6d3088e0d01d98ab45800d1b0e94712f38575 RsaQ:e
c244a18729e63a990ddcd414d0066af68c31343bd6ac03a189baa98783436d19c455fd494f5ab10ccd9ab9d18550e
ed6c2929ac0465165349c175a81eaa24da2c47de1a2dbe88eb4434e7b68b32b89fc2aec6433046713a048d54b5b86
766333d540b965c5bac6e4a971d7b804a5f39c8f7aae6f301468857d15fdf3d4c4fdb RsaN:d77af1e9b6464e6348
34e85e48969f5d649eb89fa16566a54daa95135b4b3ad8be44bf8c0c1454575059627c34ddd460b4424080e87c0c8
16550e54f9f68b6a1daeeab2d4b6da896544a3630e044f30d640830a9ab01c5ca2d77840d534a51147b6aba70a07b
3a75f76962052f2769989dc4abd6ee12eb19dc62273bddf483793cd0af625f54db606fb205e2ffa3ed8d2300b0fc6
b3e63b061fa7c7d487c960f58edfce17b0ee8c14693b3a1ace8412c09ae77592b572e2bff4fffa4e40805574704f1
6ab1aa7e66ed3d67e76a101dae09f504c1c607c1345ab17d7c16884cf80ebff2f3702d6d81472ed378f8137c2dda5
a5556c81aa5c8c31ed1a9dc3e4617 RsaE:11 RsaD:1c76beff6efefbd2fe2d8f80f64d7d6802b94ad91d826e40a2
6ec5c190f26cb1a23f812107ac07f883159511331a657fb25cc391290370e037a759bbca06f6929b33de9a75398c5
cc62e42dd81c0b84783d5c135d9d3526643d38d59350227c569dcf57d92b0607d7c5b1061e81c747453306f778963
74ead8afb4de6e29480da8b1df30a2b59a39aeb04c8118f3b2cc47f4bf1581245e8cdb687dd0b15c768de4ce74d2c
86ab16f3cf08d9d6f7b8619cb9a7a8790377d55d6600f9714836db6ad90379d35d10e5c4cc552d1ad28be125bef5b
081fe449246c612299dbc64f24ccfde6158d5bdc43c8748b5f08b82db1bc478ce408c538b398a68293e2f035 Ecds
aX:3bac7e95a003264cc075a2ba8d4e949862acd755d49094ad8d28bd0d56299dc6 EcdsaY:5c6a5b3810181d82f5
```



In this example, CDF should complain about the maximum public exponent size the Go implementation support: if we check its code we can see the public exponent is being stored as a normal integer, whereas in CryptoPP (and most other implementations), it is stored as a big integer. This is however by design and will likely not be changed.