# CERTIFICATE fuzzer – CFUZZ

**The tool creates different corruption cases for certificates that can be tested on the TLS/SSL Server. It would also give a base to work from in order to add other SSL protocol intelligence into the tool. We can find vulnerabilities in OpenSSL. It also provides the following benefits.**

## Introduction

Fuzzing is generally carried out to test security problems in software or hardware systems. It is a type of random testing. More often, file formats and network protocols are the targets of testing, still any kind of program input can be fuzzed.

The basic form of fuzzing involves sending a stream of random bits to software through command line options. This technique of random inputs serves as an efficient technique to identify bugs in network protocols, command-line applications, GUI-based applications and services. Another widely used and easy method to implement fuzzing is mutating existing input. For example, files from a test suite can be bit flipped at random or moving blocks of the file.

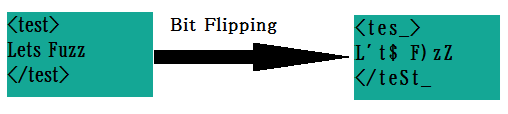


Figure 1: Fuzz Logic

We can classify fuzzing programs into two types namely mutation-based fuzzers and generation-based fuzzers. The fuzzer implemented in this case is a deterministic mutation fuzzer.

The fuzzer uses the openSSL library as its base. **OpenSSL** is an open-source software which has implemented the SSL and transport layer security (TLS) protocols. The core library is written in the C programming language and includes implementations of most of the basic cryptographic functions. It also supports many utility functions.

The result of the fuzzing process is that the tool creates a small, self-contained corpus of interesting test cases. These are very helpful for seeding other, labor- or resource-intensive testing regimes. The fuzzer is extensively tested to provide out-of-the-box performance which is superior to blind fuzzing or coverage-only tools.

Architecture of the digital certificate fuzzer tool developed is shown in Figure 4.

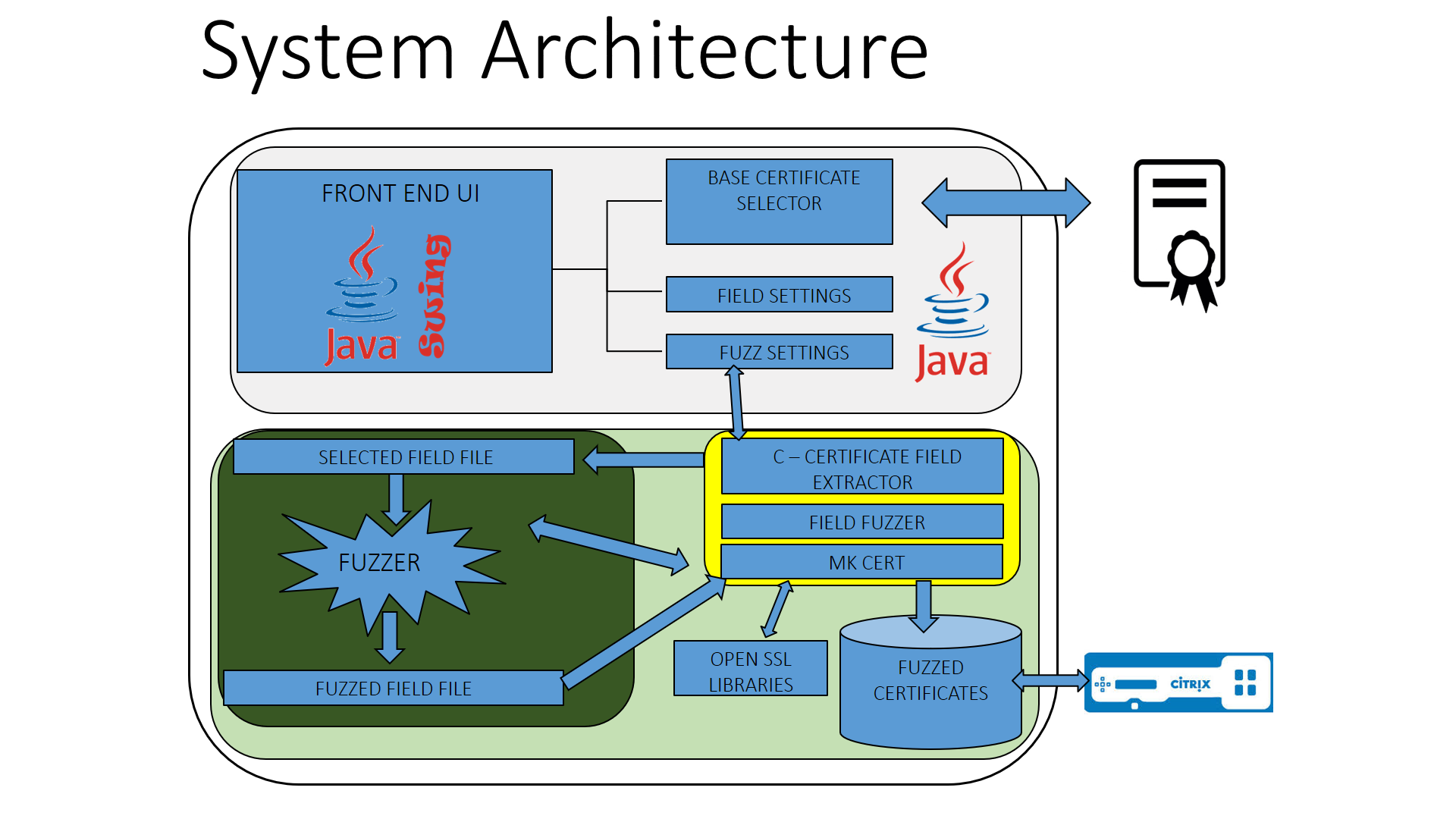
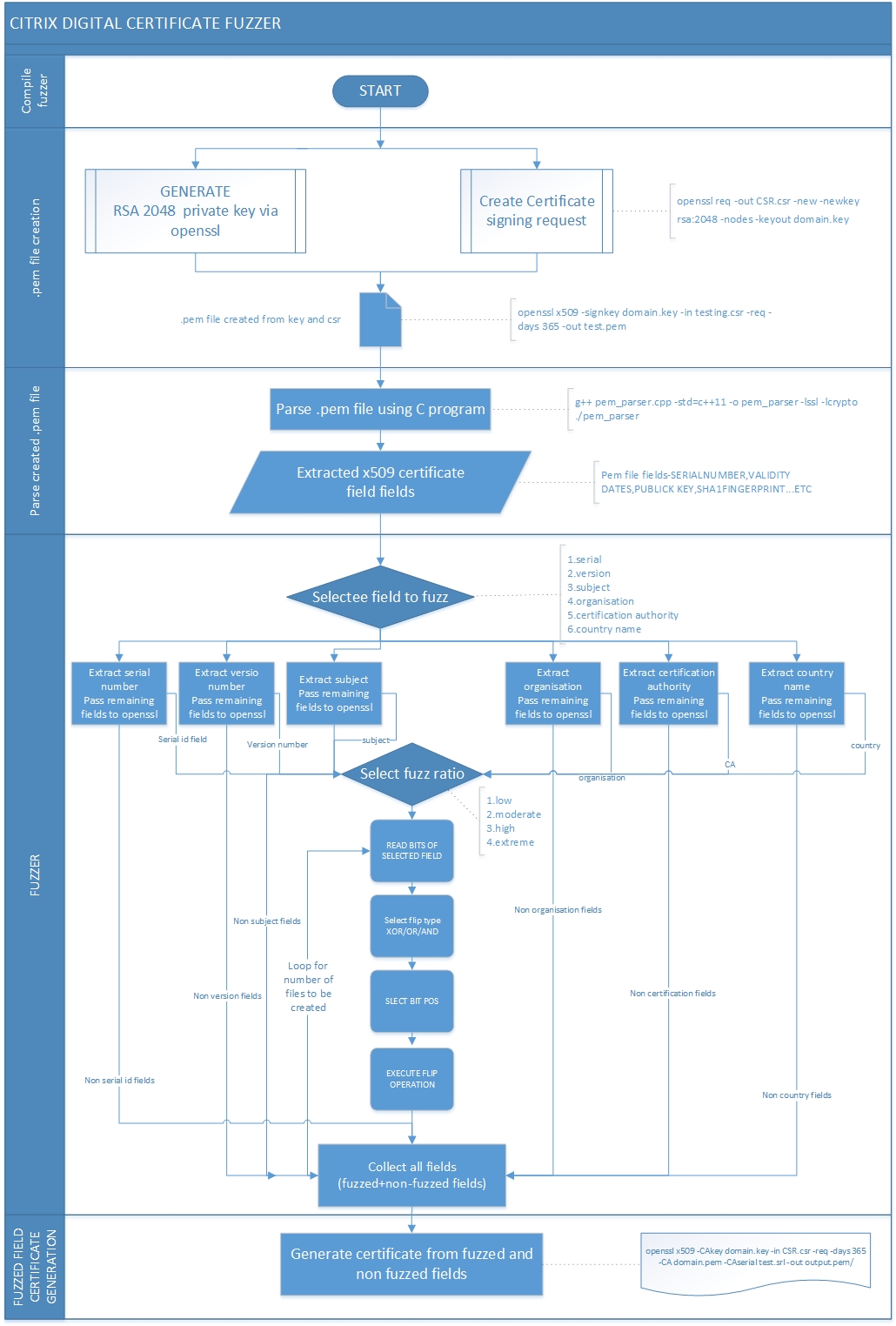
****

Figure 2: System architecture of tool

The overall working of the tool is best depicted by the flow diagram shown in Figure 3.

**

***Figure 3: Flow diagram of the digital certificate fuzzer tool***

## development OF the tool

The user interface was built using the java swing application programming interface (API) along with the JForm design tool. The interface allows for the user to interact and set parameters such as base certificate path, extent of fuzzing (predefined – low medium high extreme), number of fuzzed files to be generated and the field(s) that the tool fuzzes. A sample of GUI is shown in Figure 4a.

The Java code integrates through java native interface to the C code used to call the fuzzer and passes the relevant parameters to it. The initial parameters to the C code are obtained from the user interface and stored into a temporary intermediate file. This file is then passed as input to the fuzzer which does bit flip operations to produce more mutated files which are passed through a loop back feed into the C program.

Now depending on the level of fuzzing chosen, the fuzzer uses a ratio which determines the percentage of the file upon which bit flip operations are to be performed. For each level of fuzzing there corresponds a range of percentages are defined in 25 percent intervals corresponding to an integer value (r) between 0 to 1. The consistency of the value of r is determined using the expected average number of flipped bits with a given ratio:



The C code uses the openSSL API to generate digital certificates of various formats (.pem).

## execution method

All dependencies and helper files have been packaged within the .jar file. To execute the operations run command “*sudo java –jar cfuzz.jar*”.

The c code within the jar file have 2 prime functions-

**Certicreate.cpp**

Takes in parameters from the java UI and determines the number of fuzzed files, fuzz ratio and passes the selected field that has been chosen to be fuzzed to the fuzzer. The fuzzed temp files are then used to create certificates (.pem) using the openssl libraries in the specified folder.

**Individual.cpp**

The code takes parameters from the java UI for the various fields of a X.509 certificate and generates a key and certificate in the specified folder.

**NOTE:** on first execution of program on a new system, the logs will display “zzuf not found”, this is due to the simultaneous installation of zzuf package and running of c-code. Subsequent executions will not show same warning.

## Using the Cfuzz GUI

Below is described the sequence of steps recommended for effective use of the tool for generating fuzzed certificates using an initial base certificate

1. Double click and deploy the jar after checking all base configurations and dependencies
2. Using the file browser (labeled Select Base File) select the initial digital certificate (. pem) that is to be used as the base certificate for the mutation process.
3. Using the file browser (labeled Select Key File) select the Key value used to encryption within the certificate (.key)
4. Select from the drop down box (labeled SET FUZZ LEVEL) the required level of fuzz level (which determines the extent to which the file bits are flipped to)
   1. Low (fuzz ratio: 1)
   2. Medium (fuzz ratio: 2)
   3. High (fuzz ratio: 3)
   4. Extreme (fuzz ratio: 4)
5. Select from the spinner box (labeled SET NUMBER OF FILES) select the number of files that are to be created by the fuzzer (1 to int max).
6. From the drop down box (labeled SET FIELD) pick the desired field to be fuzzed.
   1. Serial number (default)
   2. Version
   3. Issuer subject - O
   4. Issuer subject – CN
   5. Netscape comment
   6. All (every field of the .pem certificate)
7. Using the file browser (Labeled Output Destination) select the output destination where the digital certificate(s) are to be generated and stored.
8. Click the,” START FUZZING” and wait for the progress bar to reach 100%

Note: In case of no response or need to stop fuzzing process click the “TERMINATE” button to abort the process.

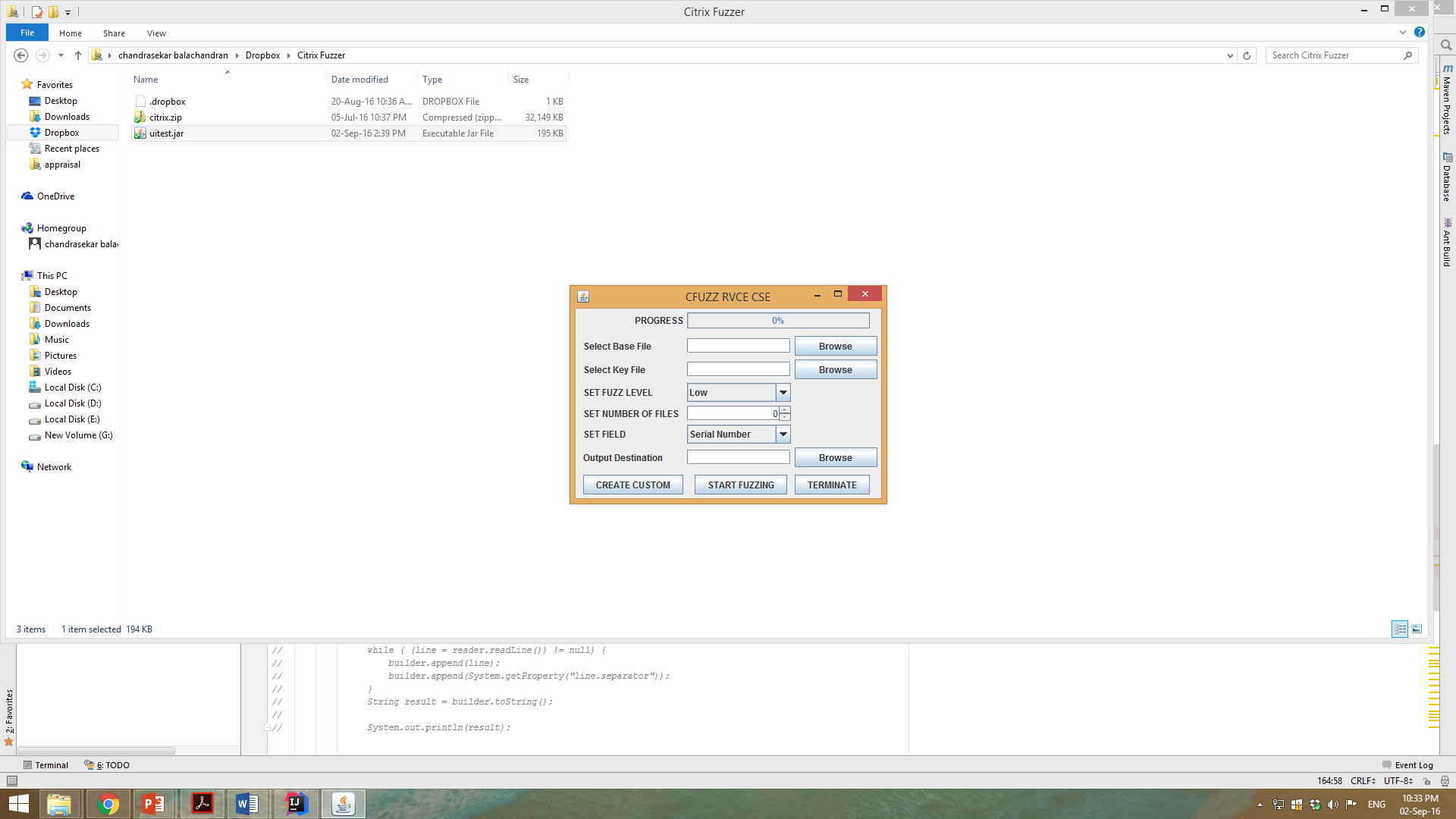


Figure 4a Main fuzzing console

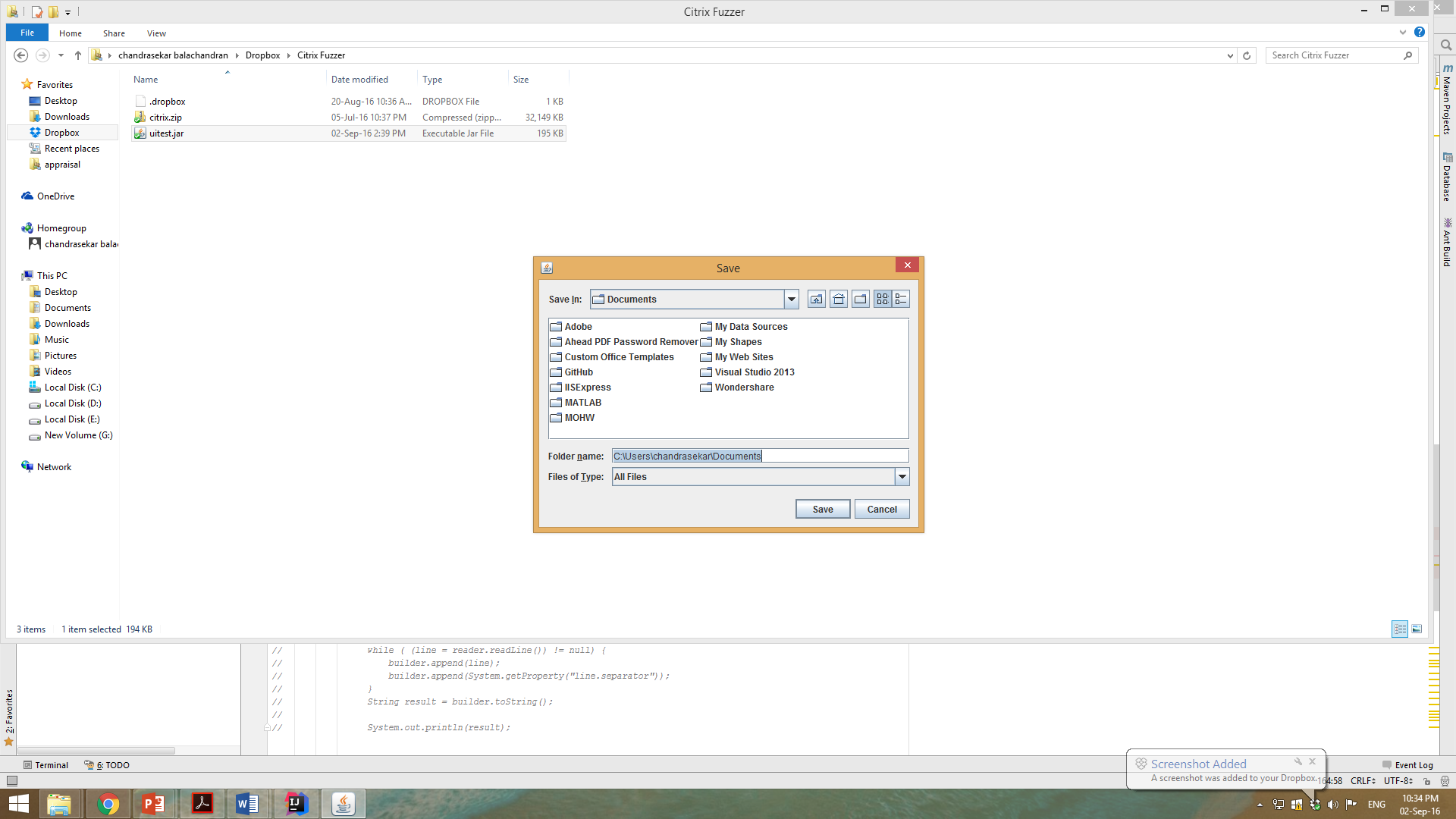


Figure 4b File browser for picking path and destination

Below is described the sequence of steps recommended for effective use of the tool to create a single custome digital certificate using the java interface and the openssl methods.

1. Same as previous
2. Click the “CRETE CUSTOM” button located at the bottom right of the form. A new window labeled “CITRIX RVCE CUSTOM CERTIFICATE CREATOR” will appear.
3. Fill the fields with the desired values
4. Using the file browser (Labeled Output Destination) select the output destination where the digital certificate is to be generated.
5. Click the “CREATE” button to begin execution and then check the output file destination for the certificate.

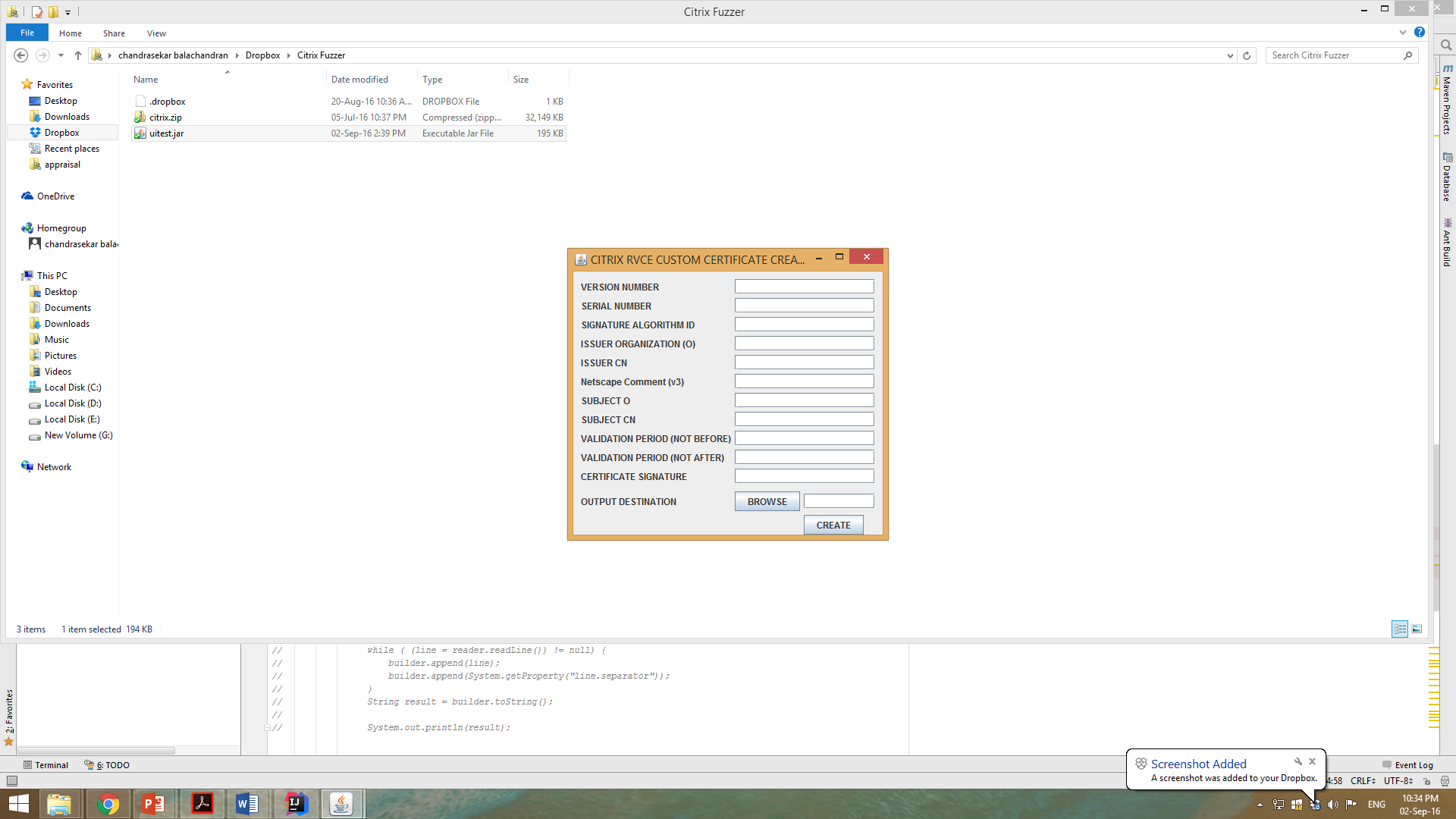
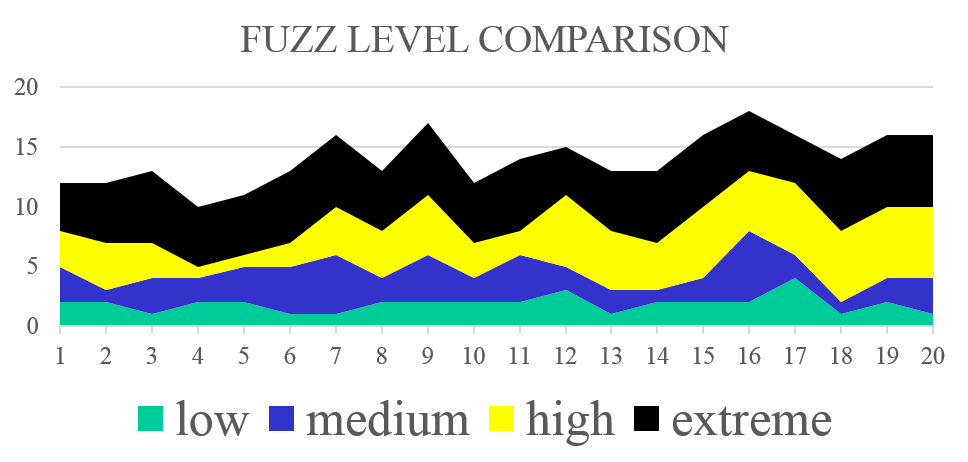


Figure 4c Custom certificate field input

## TESTING and RESULTS

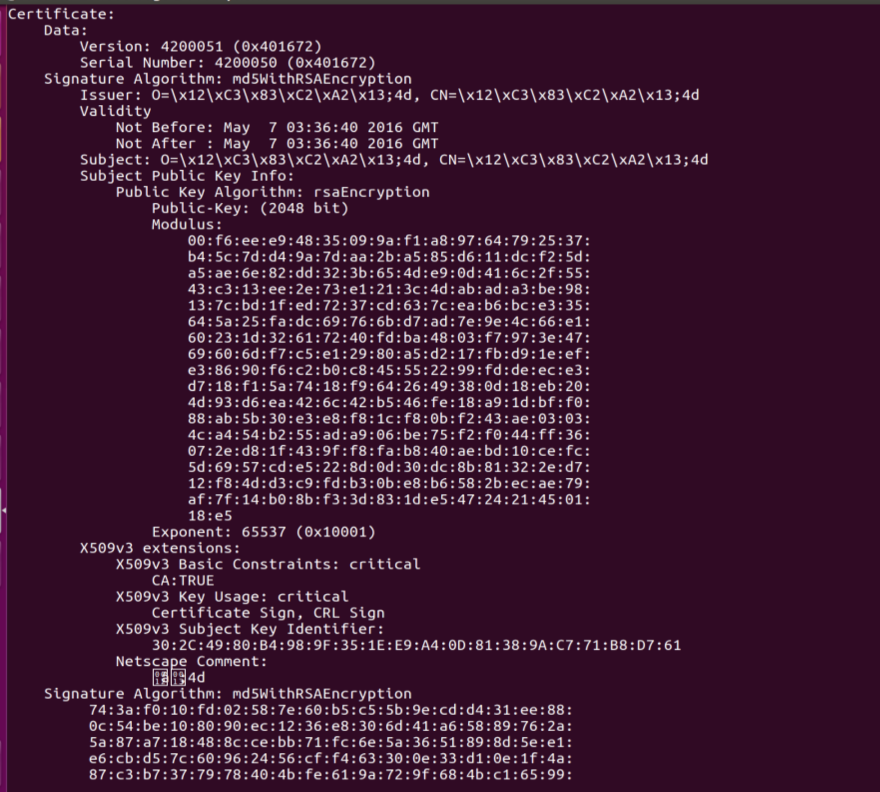
After generating several 1000 fuzzed certificates, we then compared the various fields of the certificate for different fuzz ratios to the original base certificate

The metric for comparison used is **Levenshtein distance** as shown in Figure 5. **Levenshtein distance** (LD) is measured by noting the similarity between two strings, termed as the source string and the target string. The **distance** indicates the count of insertions, deletions, or substitutions needed to transform source string to target string. [9]



***Figure 5: Comparison between fuzz levels using Levenshtein distance***

An example of a fully fuzzed extreme fuzzed level certificate for all fields is show in Figure 6.



***Figure 6: Fuzzed digital certificate***

## Salient FEATURES of the Tool

1. It is robust system capable of handling various digital certificate formats and suitable for long term compatibility.
2. The system uses an intuitive GUI that allows for implementation of fuzzing without much knowledge of fuzzing methods.
3. The tool can provide results without interaction. Once the digital certificate fuzzer is started, it can run for hours or days to identify bugs.
4. Tool is capable of finding bugs which can be missed in a manual audit process.
5. The tool is also capable of generating a large number of corruption cases of varying fuzzed levels.
6. The tool is also capable of fuzzing specific fields as mentioned by the user. This facilitates better modular testing as required by the authentication process.
7. It allows for generation of single certificate with custom user defined field values.
8. The tool is fast and is easily parallelizable.
9. The tool is compatible with all Linux distribution OSs.

## CONCLUSION

The fuzzing tool developed is a powerful strategy for identifying security issues in real-world digital certificates. The strengths of fuzzer lie within black box domain. **Black**-**box testing** is a type of software testing to examine only the functionality of an application without going into the internal details. Black-box testing method can be carried out to almost every level of software testing: unit, integration, system and acceptance.

Though the fuzzing has large test case coverage, it cannot be considered as extensive and exhaustive. Further testing is recommended after fuzzing to ensure complete security coverage of digital certificate authentication system. The field of digital certificate fuzzing is fairly new and novel. The tool thus developed is a simple and effective means of carrying out testing in the field of digital certificates.