



SOFTWARE DESIGN & IMPLEMENTATION

COSC 50 – Fall 2024

Fuzz testing

GRACE HOPPER'S LAB NOTEBOOK

9/9

0800 Andam started
 1000 " stopped - andam ✓
 1300 (032) MP - MC $\begin{cases} 1.2700 & 9.037847025 \\ 9.037846995 & \text{concord} \end{cases}$
~~1.582647000~~
 (033) PRO 2 $\begin{cases} 2.130476415 \\ 2.130676415 \end{cases}$
 concord
 Relays 6-2 in 033 failed special speed test
 in relay " 10.000 test.

Relays changed
 1100 Started Cosine Tape (Sine check)
 1525 Started Multi-Adder Test.

1545  Relay #70 Panel F
 (moth) in relay.

First actual case of bug being found.
~~1630~~ 1630 Andam started.
 1700 closed down.



TESTING

- White Box testing
 - Testing from **programmer's** perspective
 - You know how the program is designed / implemented
 - Push the bounds you know are there
 - Stress algorithms, data sizes, etc.
- Black box testing
 - Testing from **user's** perspective
 - No knowledge of design / implementation, just usage

EXAMPLE: PREVENTING DIRECTORY HOPPING

Microsoft's original webserver had URL validation scattered in several places, so when bugs were detected they were hard to fix.

`http://somedomainname.com/ ...`

- ``../..`` attack first appeared in 1996
- ``%2f`` hex code for `'/'` in 1997
- ``%2w`` (invalid hex, translated to `%2f`) in 1999
- ``%c1%1c`` (unicode for `'/'`) in 2000
- ``%252f`` which is expanded to `%2f` in 2001

SQL INJECTION

An application accepts a string from the user and then constructs a SQL query from it.

```
select * from instructor where name = '"' + inputstring + '"'
```

Everything's fine until user Eve enters the following instead of a name:

```
X' or 'Y' = 'Y
```

Then the resulting statement becomes:

```
select * from instructor where name = '"' + "X' or 'Y' = 'Y" + '"'
```

which becomes:

```
select * from instructor where name = 'X' or 'Y' = 'Y'
```

Eve could have been nice and responded with this instead:

```
X'; update instructor set salary = salary + 10000;
```

HI, THIS IS
YOUR SON'S SCHOOL.
WE'RE HAVING SOME
COMPUTER TROUBLE.



OH, DEAR - DID HE
BREAK SOMETHING?
IN A WAY -)



DID YOU REALLY
NAME YOUR SON
Robert'); DROP
TABLE Students;-- ?



OH, YES. LITTLE
BOBBY TABLES,
WE CALL HIM.

WELL, WE'VE LOST THIS
YEAR'S STUDENT RECORDS.
I HOPE YOU'RE HAPPY.



AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.

SOFTWARE TESTING - FUZZING

- Using invalid, out of range, or random data as input during program testing.
- Does the program react as you hoped?
 - Error messages to user?
 - Ignores the bad data (when possible)
- ... or does the program ...
 - Quietly run anyway, using the bad data
 - Produce bad results
 - Crash

“

THE GOAL OF FUZZ TESTING IS TO IDENTIFY ISSUES THAT CAN BE EXPLOITED BY AN ATTACKER, SUCH AS BUFFER OVERFLOWS, SQL INJECTION, OR OTHER TYPES OF INPUT-VALIDATION ISSUES.

”

Barton Miller, University of Wisconsin, 1989

<https://pages.cs.wisc.edu/~bart/fuzz/>

WHEN TO USE IT?

Can be done at any phase of testing: unit, functional, integration, etc.

- Input fuzzing
 - Command-line as well as user input
 - Strange URL's encountered by crawler's pageScan
- File fuzzing
 - Malformed / invalid / missing input files
- Network fuzzing
 - Strange responses during network communications
 - Unexpected network responses (protocol errors, authentication errors, etc.)



PROS

- Can be automated
 - Regression testing!!
- Early detection
- Expandable

CONS

- Focuses only on inputs
- Limited by tester knowledge of *valid* inputs
- Can cause crashes/hangs
 - Until you fix them!