

Software Testing, Quality Assurance & Maintenance—Lecture 1

Patrick Lam
University of Waterloo

January 7, 2019

“The Testing Course”

Course mechanics

Drop-In Hours: W 11:30-12:20, DC2539

Textbook: none

Github `git@github.com:patricklam/stgam-2019.git`

Piazza (you know where to find it)

Grace days: You may submit assignments up to 2 days late in total.

Course staff

Instructor: Patrick Lam

TAs: Meet Bhavsar (SE 2019)
 Jason Milasincic (SE 2019)
 Parsa Pourali (PhD student)
 Michael Socha (SE 2019)

Evaluation

3 individual assignments	20%
	(6 2/3% each)
Course project (up to 3/group)	15%
Midterm	15%
Final exam	50%

Midterm, final are open-book, open-notes.

Failures

Let's consider:

- consequences;
- causes;
- avoidance (before it's too late);
 - testing
- mitigation (afterwards).

A problem has been detected and windows has been shut down to prevent damage to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

Some Failures



Who suffers from failures?

Photos: (L) epicfail.com; (R) copyright ESA/CNES/ARIANESPACE - Service Optique CSG

More Failures



<http://hermosodia.wordpress.com/2008/10/19/definicion-visual-de-workaround/>



(United States Centre for Disease Control, 04MI074)



(stephen mantler at Flickr, "A runner's injury")

Infamous Software Bugs

Therac-25, 1985–1987:

5 deaths, severe injuries

race conditions, no automated testing

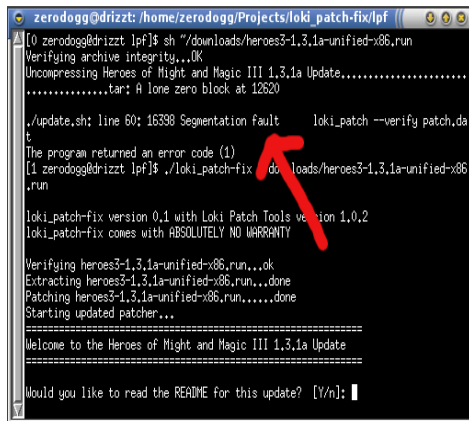
Northeast blackout, 2003

(no ice storm)

Ariane 5 crash, 1996

Morris Worm, 1988

Why Does Software Go Wrong?



A terminal window titled 'zerodogg@drizt: /home/zerodogg/Projects/loki_patch-fix/lpf'. The user runs a command to update 'heroes3-1.3.1a-unified-x86.run'. The process shows verifying the archive, extracting, and patching. A red arrow points to the error message: './update.sh: line 60: 16398 Segmentation fault loki_patch --verify patch.dat'. The terminal then shows the program returned an error code (1) and the user runs the patcher again. The patcher version is 0.1, and it shows the update progress, including a welcome message and a prompt to read the README.

```
zerodogg@drizt: /home/zerodogg/Projects/loki_patch-fix/lpf
[0 zerodogg@drizt lpf]$ sh "/downloads/heroes3-1.3.1a-unified-x86.run
Verifying archive integrity...OK
Uncompressing Heroes of Might and Magic III 1.3.1a Update.....
.....tar: A lone zero block at 12620

./update.sh: line 60: 16398 Segmentation fault loki_patch --verify patch,da
t
The program returned an error code (1)
[1 zerodogg@drizt lpf]$ ./loki_patch-fix "/downloads/heroes3-1.3.1a-unified-x86
.run

loki_patch-fix version 0.1 with Loki Patch Tools version 1.0.2
loki_patch-fix comes with ABSOLUTELY NO WARRANTY

Verifying heroes3-1.3.1a-unified-x86.run...ok
Extracting heroes3-1.3.1a-unified-x86.run...done
Patching heroes3-1.3.1a-unified-x86.run.....done
Starting updated patcher...

=====
Welcome to the Heroes of Might and Magic III 1.3.1a Update
=====

Would you like to read the README for this update? [Y/n]:
```

1. Segfaults—or crashes; infinite loops too.

Why Does Software Go Wrong?

```
public int add(int x, int y) {  
    return x - y;  
}
```

2. Wrong Output:

- method or module returns wrong information or has unwanted side effect.

Why Does Software Go Wrong?

3. Wrong API

- a library can't do what you need it to do; or
- subsystems don't work together correctly.



Photo copyright ESA/CNES/ARIANESPACE - Service Optique CSG

Why Does Software Go Wrong?

4. Bad system-level behaviour:

- Wrong output to user.
- Bad security.
- Wrong specifications.

```
chus@ATAHUALPA:~$ ./xxx
-----
Linux vmsplce Local Root Exploit
By qaaz
-----
[+] mmap: 0x0 .. 0x1000
[+] page: 0x0
[+] page: 0x20
[+] mmap: 0x4000 .. 0x5000
[+] page: 0x4000
[+] page: 0x4020
[+] mmap: 0x1000 .. 0x2000
[+] page: 0x1000
[+] mmap: 0xb7d72000 .. 0xb7da4000
[+] root
root@ATAHUALPA:~# id
uid=0(root) gid=0(root) grupos=20(dialout),24(cdrom),25(floppy),29(audio),
44(video),46(plugdev),106(netdev),109(powerdev),1000(chus)
root@ATAHUALPA:~#
```

Why Does Software Go Wrong?

- 5. Nonfunctional properties:
 - Leaks (yes, even in Java).
 - Performance.

Why Does Software Go Wrong?

Regressions to past bugs.

Avoiding Software Failures

- test the software (in-house, externally)
- require validation suites for plugins
- code review
- better design (“write better code!”)
- include fewer features
- defensive programming
(especially for plugins)

Mitigation: Failure is Inevitable

Software never completely works.

Aim: make software that is good enough.

Coping with an Imperfect World

- disclaim liability

25. LIMITATION ON AND EXCLUSION OF DAMAGES. You can recover from Microsoft and its suppliers only direct damages up to the amount you paid for the software. You cannot recover any other damages, including consequential, lost profits, special, indirect or incidental damages.

(Vista license)

Coping with an Imperfect World

- disclaim liability
- release patches
- backup/replicate user data
- defensive programming

Ways of Testing Software

- compile it

Ways of Testing Software

- compile it
- run it on one input

Ways of Testing Software

- compile it
- run it on one input
- run it on many inputs

Ways of Testing Software

- compile it
- run it on one input
- run it on many inputs
- run it on a representative set of inputs

Ways of Testing Software

- compile it
- run it on one input
- run it on many inputs
- run it on a representative set of inputs
- run it on all inputs (static analysis)

Other Testing Concerns

- Integration testing
- Nonfunctional properties
- Regression tests

Part I

About This Course

Goals of This Course

- You will be able to create and evaluate test suites for reasonably-sized software systems.
- You will learn how to use and write tools for software maintenance and verification (particularly automated testing tools).

Part I: Defining Test Suites

Key questions:

- find interesting inputs;
- know when to stop looking.

Generating Test Suites

We'll see:

- open-ended exploratory testing;
- statement/branch coverage;
- graph-based models of program state;
integrating design documentation;
- automatically generating inputs: grammars,
fuzzing.

Evaluating Test Suites

How good is your test suite?

- coverage
- mutation

Part II: Engineering Test Suites

What are best (and worst) practices for implementing test suites?

- xUnit
- web-based testing
- mock objects
- refactoring tests
- refactoring code to be testable
- continuous integration
- flaky tests

Part III: Tools for Verification & Validation

What's out there beyond testing?

- concept: static vs. dynamic analysis
- static approaches
- dynamic approaches
- human-based approaches: code review, bug reporting

Bonus: Debugging and the Scientific Method

Don't: randomly debug your code.

Do: Make hypotheses and verify them.

Reference: Andreas Zeller. *Why Programs Fail: a Guide to Systematic Debugging*.

Part II

Defining some terms

Terminology

Validation: evaluating software prior to release to ensure compliance with intended usage.

Verification: determining whether products of a given phase of the development process fulfill requirements established in a previous phase.

Terminology

Software fault: static defect in the software.

Software error: incorrect internal state that is the manifestation of some fault.

Software failure: External, incorrect behaviour (as in “epic fail”).

Testing vs. debugging

Testing:

evaluating software by observing its execution.

Debugging:

finding (and fixing) a fault given a failure.

Problems with Manual Testing

- Testing tasks are often repetitive (i.e. boring).
- It is easy to make mistakes while carrying out tests.

Automation

Automation is key to successful testing:

- Enables you to run more tests more quickly.
- Helps ensure coverage.

Designing test suites is real engineering
(in the broad sense).