SCIENCE
PASSION
TECHNOLOGY

Real-World Study of the Security of Educational Test Systems

Stefan Gast[1] Sebastian Daniel Felix[1] Alexander Steinmaurer[2] Jonas Juffinger[1] Daniel Gruss[1]

[1]Graz University of Technology
[2]Interdisciplinary Transformation University Austria

OSVS 2025

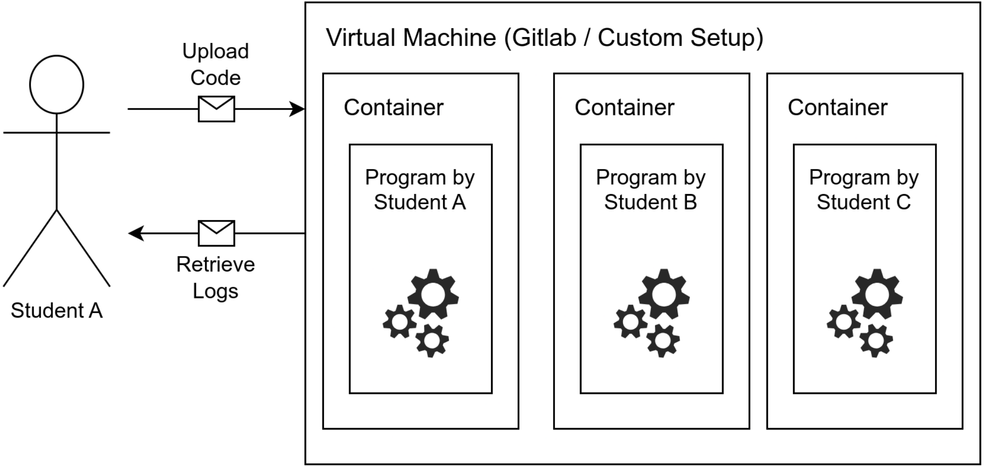
Educational Test Systems

Do you teach programming?

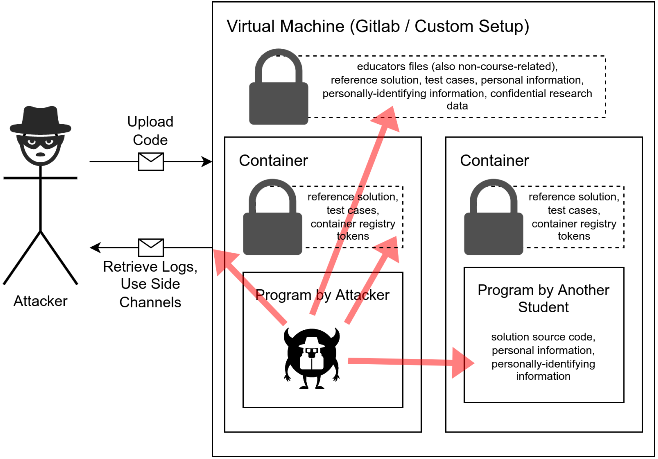

EXAM
EXAM
EXAM
EXAM
EXAM
EXAM

Do you use a test system for grading?

Test System Setup



Threat Model



Analyzed Test Systems

System	Educational Context	Languages	User Interface	Students
GD1	Introductory	C	Git	770
GD2	Introductory	C	Git	366
GD3	Introductory	C++	Git	547
C4	Systems	C, C++, Assembly	Git, Web	134
GD5	Formal Methods	Python	Git	125
GD6	Privacy	C++, Python	Git	57
GC7	Software Development	Java	Git	413
C8	Systems	C++	-	547
GD9	Security	Python	Git	413
C10	Systems	C, C++, Assembly	Git, Web	374
C11	Security	C	Web	75

Analyzed Test Systems

System	Educational Context	Languages	User Interface	Students
GD1	Introductory	C	Git	770
GD2	Introductory	C	Git	366
GD3	Introductory	C++	Git	547
C4	Systems	C, C++, Assembly	Git, Web	134
GD5	Formal Methods	Python	Git	125
GD6	Privacy	C++, Python	Git	57
GC7	Software Development	Java	Git	413
C8	Systems	C++	-	547
GD9	Security	Python	Git	413
C10	Systems	C, C++, Assembly	Git, Web	374
C11	Security	C	Web	75

GD: GitLab Runners with Docker, **GC:** GitLab Runners with custom pipeline, **C:** fully custom

Methodology

Asset-oriented Security Evaluation

Primary Interest Assets

- private test cases
- reference solutions
- other students' solutions
- other students' scores

Asset-oriented Security Evaluation

Primary Interest Assets

- private test cases
- reference solutions
- other students' solutions
- other students' scores

Secondary Interest Assets

- credentials / access tokens
- code execution as root
- container escape

Asset-oriented Security Evaluation

Primary Interest Assets

- private test cases
- reference solutions
- other students' solutions
- other students' scores

← Intermediate Step

Secondary Interest Assets

- credentials / access tokens
- code execution as root
- container escape

Continuous Integration for the Attacker

Benign students...

Attackers...

Continuous Integration for the Attacker

Benign students...

- get feedback about their solutions quickly

Attackers...

Continuous Integration for the Attacker

Benign students...

- get feedback about their solutions quickly

Attackers...

- get feedback about their exploits quickly

Continuous Integration for the Attacker

Benign students...

- get feedback about their solutions quickly
- often can write output to log file

Attackers...

- get feedback about their exploits quickly

Continuous Integration for the Attacker

Benign students...

- get feedback about their solutions quickly
- often can write output to log file

Attackers...

- get feedback about their exploits quickly
- can exfiltrate secrets via log file

What if attackers cannot write to the log file?

- use other pipeline artifacts (if possible)

What if attackers cannot write to the log file?

- use other pipeline artifacts (if possible)
- send to attacker controlled server

What if attackers cannot write to the log file?

- use other pipeline artifacts (if possible)
- send to attacker controlled server
  - Disable or filter Internet Access!

What if attackers cannot write to the log file?

- use other pipeline artifacts (if possible)
- send to attacker controlled server
  - Disable or filter Internet Access!
- use timing covert channel

What if attackers cannot write to the log file?

- use other pipeline artifacts (if possible)
- send to attacker controlled server
  - Disable or filter Internet Access!
- use timing covert channel

For every secret byte `b` do:

```
sleep(b);
```

Categorization of Security Issues

Output Sanitation

Permissions

Misconfiguration

Environment Sanitation

Categorization of Security Issues

Output Sanitation

- failure to block attempts to output sensitive information
- hard due to arbitrary encodings, side channels

Permissions

Misconfiguration

Environment Sanitation

Categorization of Security Issues

Output Sanitation

- failure to block attempts to output sensitive information
- hard due to arbitrary encodings, side channels

Permissions

- failure to block access to sensitive resources
- e.g. file permissions, process capabilities

Misconfiguration

Environment Sanitation

Categorization of Security Issues

Output Sanitation

- failure to block attempts to output sensitive information
- hard due to arbitrary encodings, side channels

Permissions

- failure to block access to sensitive resources
- e.g. file permissions, process capabilities

Misconfiguration

- e.g. unfiltered Internet access, outdated software, unprotected access tokens

Environment Sanitation

Categorization of Security Issues

Output Sanitation

- failure to block attempts to output sensitive information
- hard due to arbitrary encodings, side channels

Permissions

- failure to block access to sensitive resources
- e.g. file permissions, process capabilities

Misconfiguration

- e.g. unfiltered Internet access, outdated software, unprotected access tokens

Environment Sanitation

- failure to protect build/test environment
- e.g. `Makefile`, `.gitlab-ci.yml`

Results

Mitigated ✓ and Unmitigated ✗ Attack Vectors

Security Issue	GD1	GD2	GD3	C4	GD5	GD6	GC7	C8	GD9	C10	C11
Unsanitized Output	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
Timing Side Channels	✗	✗	✗	?	✗	✗	?	?	✗	?	?

Mitigated ✓ and Unmitigated ✗ Attack Vectors

Security Issue	GD1	GD2	GD3	C4	GD5	GD6	GC7	C8	GD9	C10	C11
Unsanitized Output	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
Timing Side Channels	✗	✗	✗	?	✗	✗	?	?	✗	?	?
No Container	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Privileged Git Tokens	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓
Reference Solution	✗	✗	✗	✓	✗	✓	✓	✗	✓	✓	✓
Exec as Root	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗

Mitigated ✓ and Unmitigated ✗ Attack Vectors

Security Issue	GD1	GD2	GD3	C4	GD5	GD6	GC7	C8	GD9	C10	C11
Unsanitized Output	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
Timing Side Channels	✗	✗	✗	?	✗	✗	?	?	✗	?	?
No Container	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Privileged Git Tokens	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓
Reference Solution	✗	✗	✗	✓	✗	✓	✓	✗	✓	✓	✓
Exec as Root	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗
`DOCKER_AUTH_CONFIG`	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓	✓
`ptrace`	✓	✓	✓	✗	✗	✗	?	✗	✓	✗	✓
Internet Access	✗	?	?	✗	✓	✓	✗	✗	✓	✓	✓
Outdated Software	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Docker Socket Mount	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓

Mitigated ✓ and Unmitigated ✗ Attack Vectors

Security Issue	GD1	GD2	GD3	C4	GD5	GD6	GC7	C8	GD9	C10	C11
Unsanitized Output	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
Timing Side Channels	✗	✗	✗	?	✗	✗	?	?	✗	?	?
No Container	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓
Privileged Git Tokens	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓
Reference Solution	✗	✗	✗	✓	✗	✓	✓	✗	✓	✓	✓
Exec as Root	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗
`DOCKER_AUTH_CONFIG`	✗	✗	✗	✓	✗	✗	✓	✓	✗	✓	✓
`ptrace`	✓	✓	✓	✗	✗	✗	?	✗	✓	✗	✓
Internet Access	✗	?	?	✗	✓	✓	✗	✗	✓	✓	✓
Outdated Software	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓
Docker Socket Mount	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Modify Pipeline	✗	✗	✗	✓	✓	✓	✗	✓	✓	✓	✓
Build System Takeover	✗	✗	✗	✗	✓	?	✓	✓	✓	✓	✓

Summary

Out of 11 test systems:

- 6 without output sanitization

Summary

Out of 11 test systems:

- 6 without output sanitization
- 6 exposed container registry credentials to student code via `DOCKER_AUTH_CONFIG`

Summary

Out of 11 test systems:

- 6 without output sanitization
- 6 exposed container registry credentials to student code via `DOCKER_AUTH_CONFIG`
- 5 exposed reference solutions to the attacker

Summary

Out of 11 test systems:

- 6 without output sanitization
- 6 exposed container registry credentials to student code via `DOCKER_AUTH_CONFIG`
- 5 exposed reference solutions to the attacker
- 4 allowed to manipulate the pipeline configuration

Summary

Out of 11 test systems:

- 6 without output sanitization
- 6 exposed container registry credentials to student code via `DOCKER_AUTH_CONFIG`
- 5 exposed reference solutions to the attacker
- 4 allowed to manipulate the pipeline configuration
- 5 with writable Makefiles or build scripts

Case Study GD5

Properties

Educational Context:	formal methods
Architecture:	GitLab runners + private Docker registry
Primary Assets:	test cases, reference solution
Students:	125

Case Study GD5: Findings (1/2)

Case Study GD5: Findings (1/2)

Output Sanitiation

- no sanitation
- auth. token leakable within 156 s to 1820 s via timing side channel

Case Study GD5: Findings (1/2)

Output Sanitation

- no sanitation
- auth. token leakable within 156 s to 1820 s via timing side channel

Permissions

- reference solution in container
- private testcases in container
- student code executed as root in container

Case Study GD5: Findings (2/2)

Case Study GD5: Findings (2/2)

Misconfiguration

- code can `ptrace` parent process to access test cases
- `DOCKER_AUTH_CONFIG` exposed to student code

Case Study GD5: Findings (2/2)

Misconfiguration

- code can `ptrace` parent process to access test cases
- `DOCKER_AUTH_CONFIG` exposed to student code

Environment Sanitation

- no issues found

Case Study GD1

Properties

Educational Context:	introductory
Architecture:	GitLab runners + private Docker registry
Primary Assets:	private test cases, reference solution
Students:	770

Case Study GD1: Findings (1/2)

Case Study GD1: Findings (1/2)

Output Sanitation

- no sanitation
- auth. token leakable, live reports on test-case run times

Case Study GD1: Findings (1/2)

Output Sanitation

- no sanitation
- auth. token leakable, live reports on test-case run times

Permissions

- API token for private test case repo accessible
- test case repo also contains reference solution

Case Study GD1: Findings (2/2)

Case Study GD1: Findings (2/2)

Misconfiguration

- `DOCKER_AUTH_CONFIG` exposed to student code
- outbound connections allowed when fetching test cases

Case Study GD1: Findings (2/2)

Misconfiguration

- `DOCKER_AUTH_CONFIG` exposed to student code
- outbound connections allowed when fetching test cases

Environment Sanitation

- `.gitlab-ci.yml` overwritable by students
- `Makefile` overwritable by students

Case Study C4

Properties

Educational Context:	Systems
Architecture:	custom (build in chroot, execute in Qemu fork)
Primary Assets:	private test Cases, other students' solutions
Students:	134

Case Study C4: Findings (1/2)

Case Study C4: Findings (1/2)

Output Sanitation

- some sanitation, but not during build and startup
- test cases leakable

Case Study C4: Findings (1/2)

Output Sanitation

- some sanitation, but not during build and startup
- test cases leakable

Permissions

- no issues found

Case Study C4: Findings (2/2)

Case Study C4: Findings (2/2)

Misconfiguration

- outbound connections allowed during all pipeline phases
- Qemu fork outdated and unpatched against escape vulnerabilities
- software within `chroot` outdated (escape possible via `sudo`)
- same user account for all pipelines of different students
- `ptrace` across different processes not prohibited

Case Study C4: Findings (2/2)

Misconfiguration

- outbound connections allowed during all pipeline phases
- Qemu fork outdated and unpatched against escape vulnerabilities
- software within `chroot` outdated (escape possible via `sudo`)
- same user account for all pipelines of different students
- `ptrace` across different processes not prohibited

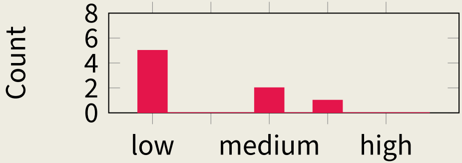
Environment Sanitation

- `Makefile` overwritable by students
- helper programs overwritable

User Study

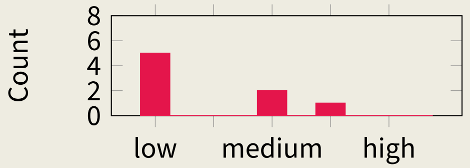
How Do Test System Maintainers Assess The Severity?

Reading Solutions of Others

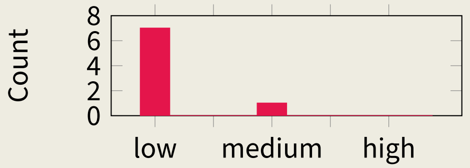


How Do Test System Maintainers Assess The Severity?

Reading Solutions of Others

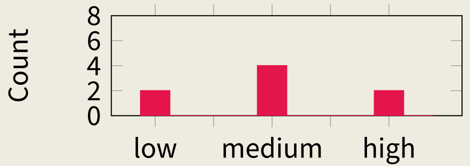


Writing Solutions of Others



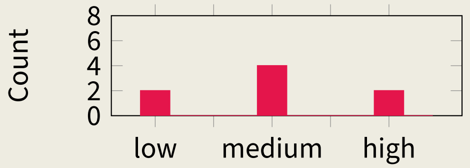
How Do Test System Maintainers Assess The Severity?

Reading Test Cases

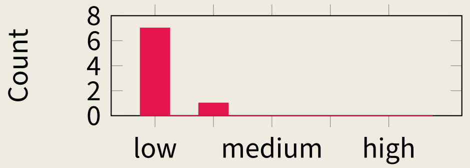


How Do Test System Maintainers Assess The Severity?

Reading Test Cases

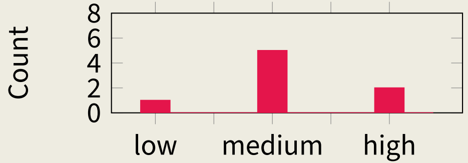


Writing Test Cases



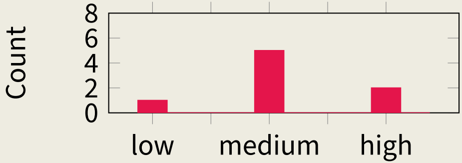
How Do Test System Maintainers Assess The Severity?

Reading All Test Container Files

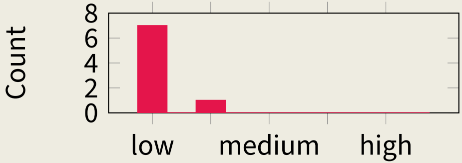


How Do Test System Maintainers Assess The Severity?

Reading All Test Container Files

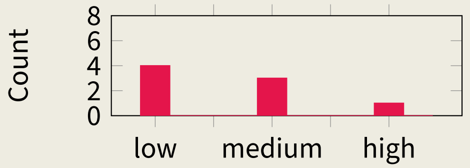


Writing All Test Container Files



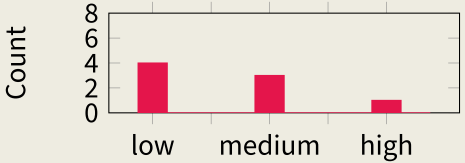
How Do Test System Maintainers Assess The Severity?

Reading Container Registry

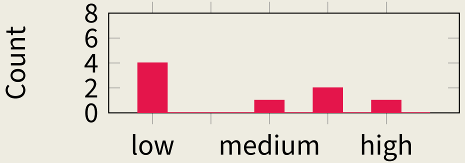


How Do Test System Maintainers Assess The Severity?

Reading Container Registry

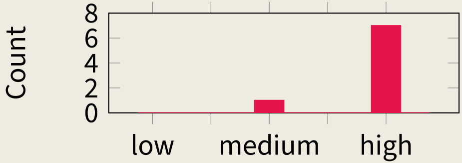


Writing Container Registry



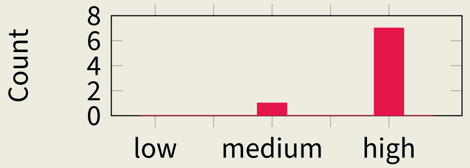
How Do Test System Maintainers Assess The Severity?

Reading Container Host

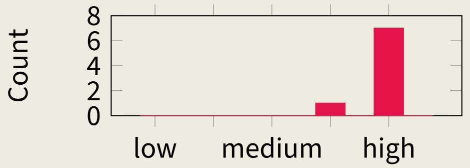


How Do Test System Maintainers Assess The Severity?

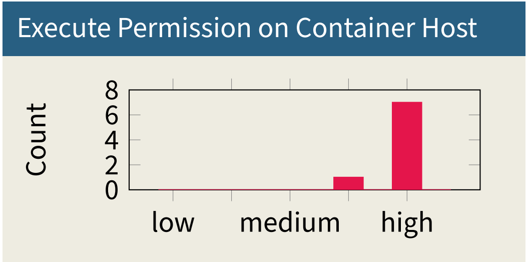
Reading Container Host



Writing Container Host



How Do Test System Maintainers Assess The Severity?



Acknowledgements

This research was made possible by generous funding from:

Funded by the European Union

fssec erc

European Research Council

Established by the European Commission

SPyCoDe FWF

Der Wissenschaftsfonds.

codeAbility [austria]

Red Hat

Google

Supported in part by the European Research Council (ERC project FSSec 101076409), and the Austrian Science Fund (FWF SFB project SPyCoDe 10.55776/F85). Additional funding was provided by generous gifts from Red Hat and Google. Some results are from the Codeability Austria Project, funded by the Austrian Federal Ministry of Education, Science and Research (BMBWF). Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties.

SCIENCE
PASSION
TECHNOLOGY

Real-World Study of the Security of Educational Test Systems

Stefan Gast[1] Sebastian Daniel Felix[1] Alexander Steinmaurer[2] Jonas Juffinger[1] Daniel Gruss[1]

https://stefangast.eu
stefan.gast@tugraz.at
notbobbytables@infosec.exchange

OSVS 2025