Build-it Break-it Fix-it Judging Rules

There are some key parts of the contest that require human judgment. This document provides specific instructions about how that judgment is to be exercised, so that both our judging volunteers and participants know what's going on.

Rules regarding Obfuscation

Background: Obfuscating code is the act of making it appear to be more complicated than necessary in an effort to make it difficult for one viewing the code to understand what it does.

Rule: Build-it teams should not obfuscate any part of their committed source code.

Consequences of rule violation: Build-it teams will be ***disqualified*** if they are found to obfuscate *any part* their code in such a way that is both:

1. Done with the **intent** to deceive the true meaning of the code, *and*
2. **Detrimental** to a break-it team's understanding of the code.

Examples of obfuscation

- Unnecessary complexity in the code (e.g., branches, loops, extra methods)
- Misleading variable names
- Use of code minimization or obfuscation tools
- Submitting only binaries, not source code (except for libraries for which source is unavailable)

Role of Judges: Judges will examine submitted code and determine whether it has been obfuscated. Judges deeming a team to have obfuscated their code will provide a brief justification for their decision. The accused team will have 24 hours to respond. A final decision will be made prior to the start of the next round.

Role of Other Teams: Any team (either a build-it or break-it team) that believes that a build-it team has obfuscated their code may submit an obfuscation report to the judges. This report should consist of:

- Name of team who appears to have obfuscated their code.
- The specific portions of the code that have been obfuscated.
- Justification of the claim that the code is obfuscated.

The team submitting the report remains anonymous to all other teams. If a build-it team is disqualified for obfuscating their code, break-it teams will still obtain full points for any bugs found in that team's code. Also, any build-it team that is disqualified for obfuscating their code will also not be allowed to participate as a break-it team.

Rules regarding Submission of Bug Reports

Background: Break-it teams will submit test cases that they claim expose bugs or vulnerabilities in the target build-it team's program. These test cases will be accompanied by a report that claims why test case reveals a problem.

Role of Contest Organizers: For correctness- and vulnerability-related bugs, we (the contest organizers) will run a "Bayesian tester" to assess the likelihood that the submitted test case actually exposes a true bug or vulnerability, as opposed to the test case itself being buggy. The Bayesian tester is a tool that automatically estimates the validity of a test case. It works by running the test case against all builder programs (and our reference solutions) and computing the likely validity via Bayesian statistics. Rather than return a simple yes/no, the Bayesian tester reports a likelihood of whether the test case is valid. The validity of all test cases will be determined by the following rules:

- Any test cases with a likelihood at or above 85% is automatically deemed valid.
- Any test cases with a likelihood at or below 40% is automatically deemed invalid.

Role of Judges: All other test cases (likelihood between 40 and 85) will be manually inspected and decided upon by at least one judge. Judges will use the following guidelines and definitions to determine the validity of a test case:

- A test case should be considered valid if it exposes a bug, vulnerability, or security violation in the target program.
    - A bug is defined as incorrect behavior by a program like crashes or deviation from the problem specification.
    - A vulnerability is defined as a security exploit in a program like SQL injection, buffer overflow attacks, etc. This will manifest as correctness-related issue (doing the wrong thing) but with consequences to security (e.g., it could facilitate remote code injection).
    - A security violation is a violation of either privacy or integrity, and its definition depends on the project specification, but will be defined so that such violations are evident by the test cases themselves; i.e., test cases will be self-evidently correct.

Judges will be presented with anonymized bug reports (i.e., the break-it team that submitted the report not be revealed).

Rules regarding Fixing Multiple Bugs with a "Single Fix"

Background: During the Fix-it phase, build-it teams need to submit fixes for single bugs, and not try to group fixes to multiple bugs in one check-in. They are supposed to submit a report that claims how they are fixing one, and only one, bug. A single fix is meant to fix a single bug in the code—in applying the fix, the bug goes away.

What is a "single fix"? A "single fix" is a modification to the code base that solves a single logical flaw in the code.

Examples of a "single fix"

- Modifying the value of a constant
- Changing UDP sockets to TCP sockets

Note that a single fix may involve multiple changes to the code. Additionally, if that single logical flaw is replicated multiple times throughout the code, then repairing all of them is still considered a "single fix", e.g., a bounds error, having "<=" instead of "<" in multiple for loops, but involving the same data.

If a single fix resolves multiple correctness-related test cases (i.e., all previously failed but now pass), then those test cases are considered equivalent. As a result, the team should only lose points for one of those test cases, not all of them. Likewise, when multiple test cases submitted by the same break-it team are considered equivalent, points will only be awarded for one of them. When test cases submitted by multiple teams are considered equivalent, they must share the points for that bug.

Role of Judges: Judges will use reports at their discretion to determine if the change constitutes a fix for a single bug or multiple bugs. If a build-it team submits a fix that resolves multiple test cases, but the judges deem that the test cases are not equivalent, then the build-it team will still lose points for all of the test cases in the report. On the other hand, if the judges agree that this a single fix, then the build-it team will get back the points for each of the duplicate test cases, and likewise, the points awarded to the break-it team or teams that submitted these test cases will be reduced.

Example Scenario:

- Report says "Test cases A, B, and C are the same, and here is one fix that addresses all of them."

- There are two possible outcomes. If the judges agree, then the build-it team gets back points for two of the test cases, and the break-it teams that submitted A, B, and C only get points for one bug (shared among them). Else the judges do not agree, believing that A, B, and C are not equivalent. In this case, the build-it team gains back no fix-it points, and the break-it teams get full marks for each test case.

Rules regarding Report Structure

Background: Fixing the basic structure of all submitted reports (by both types of teams), their maximum length, and format will help judges process the reports efficiently.

Fix-it Report: When submitting a fix during the Fix-it phase, build teams will use a template that consists of:

- A short (no more than 250 words) description of the logical flaw.
- A short (no more than 250 words) description of the fix.
- (Optional) Assuming this fixes more than one submitted test case, then a short (no more than 250 words) description of why these test cases should be considered equivalent.
- A patch file (which includes a list of all the changes to the code).

Bug Report: When submitting a claimed bug, break-it teams will provide a test case and, optionally, a bug report, which should use the following template:

- A short (no more than 250 words) description of the flaw and why it is a bug. The description may refer to relevant sections of the spec.

Rules regarding Judging Timeline

Judges will look at fix reports as soon as they are filed, but will not rule on any of them until after the fix-it round is completed. Once all of the decisions have been made, the final scores will be posted, and the winners announced.