基於沙盒系統之程式評測應用

專題編號： 112-CSIE-S001-MID
執行期限： 111 年第 1 學期至 112 年第 1 學期
指導教授： 郭忠義
專題參與人員： 109590027 溫紹傑
109590031 黃漢軒
109590033 吳秉宸

一、 摘要

NuJudger 是一個基於 Python Flask 的服務，以資訊安全、方便性與穩定考量如何設計一個程式評測系統的評測服務，保留提供給予使用者的錯誤提示並同時守護評測機敏資訊不被惡意程式外洩。我們將在這個專題中設計一套評測機制來減少答案的出錯可能並掌握因不當使用而造成服務崩潰的議題，並同時考量使用 Docker 佈署評測服務、使服務具有良好的單元測試覆蓋率、使用 GitHub Actions 持續整合（CI）確認提交通過測試與通過靜態型別檢查。

關鍵字：程式評測服務、隱蔽通道攻擊、沙盒、微服務

二、 緣由與目的

隨著程式評測系統在程式教學與程式競賽中佔有重要的地位，其資安也須受到重視。以 ZeroJudge 現有的資安漏洞能夠透過 stderr 通道將測試資料偷出，以及 NTUT Jenkins Judge 能夠在執行時透過惡意指令來印出測資檔案內容，對程式競賽與程式教學帶來了非常大的作弊風險。防禦測資被偷出的作弊問題主要發生在評測平台提供了程式錯誤相關的提示供給使用者進行除錯，關閉了提示功能雖可以解決測資被偷出的問題，但會對於使用者造成莫大的不便。

我們希望可以開發 NuJudger 評測機，這個評測機主要著重在以下幾點：

- 穩定的評測機制。
- 保證資安問題。
- 保留使用者的方便性。
- 以 HTTP POST 進行溝通，使用 Webhooks 回傳評測結果，易於串接現有的評測平台。
- 支援叢集評測。

為了能夠解決市面上並沒有穩定的評測機，我們希望可以實踐具有這些特性的評測機，並命名為 NuJudger，希望可以使程式評測具有安全與穩定的保障，並提供一個值得信賴、穩定且好用的程式評測服務。

三、 研究範圍

為了探討一個良好的評測機必須具備的條件，我們希望可以將這個專題分成幾個議題：

1) 如何確保題目的答案是有來源保證的，而非人工撰寫，使得因人為因素導致題目解答不正確。
2) 如何確保評測機制能夠涵蓋所有的評測異常情況，導致因評測異常而服務停止。
3) 如何確保評測機能夠防禦使用者的惡意程式攻擊（例如執行惡意指令）或使用特定手段偷竊測試資料，使得評測服務停止或測試資料外洩。
4) 當評測回傳結果為程式不正確時，如何揭露足夠多的錯誤資訊使得使用者能夠瞭解程式出現的問題，又能避免錯誤資訊出現測資外洩的可能。
5) 許多評測服務都具有佇列凍結（Frozen queue）的問題，主要發生在提交高峰時評測過慢導致提交排隊的問題，我們想要使用叢集評測的方法，使得評測佇列凍結的情況大幅降低。

藉由解決這一系列的問題以及良好穩定的開發，我們能夠創造一個安全、穩定且使用者方便的評測服務。

四、 使用技術方法

評測機除了可以運作正常評測程式，我們同時保證了測資外洩的防禦與叢集評測，以下我們將根據評測機制、評測機防禦、測資防禦與叢集評測來介紹這個專題的亮點。

(一) 評測機制

NuJudger 使用自設計的評測機制，使答案的來源由出題者提供的解答程式來保證。

附錄的 Figure 1 說明了當前 NuJudger 預期的評測機制。

評測機主要蒐集了提交者的程式碼（Submission）、出題者的解答程式碼（Solution）、用於判定執行結果是否正確的確認程式碼

（Checker）與測試資料（Test Case），透過提交者的程式碼與測試資料得到使用者輸出（Output），透過出題者的程式碼與測試資料得到解答輸出（Answer），最後使用確認程式來確認使用者輸出與解答輸出是否正確。由此評測機制可以保證解答輸出的來源，使人為手打錯誤導致解答錯誤的可能性降低，並且易於其他出題者測試解答程式是否正確。

(二) 評測機防禦

NuJudger 中所有的從服務皆為沙盒服務，沙盒服務為基於 IOI 開發的 Isolate 所開發的網路應用程式（Web Application），沙盒能夠限制程式的執行時間（Time Limit）、記憶體用量（Memory Limit）、檔案開啟（Open Files）、是否能夠連接外網（Share Net）與權限控管等，使得沙盒服務能夠阻擋除了隱蔽通道攻擊以外的惡意程式攻擊。

(三) 叢集評測

由於 NuJudger 中的主從服務主要皆是使用 HTTP POST 進行資料的傳遞，利用這樣的方式，我們可以使從服務從內網跨至外網，僅須保證網路連通與測資是同步的。使用這樣的方式即可由主服務控管由從服務完成評測的評測結果，並傳回伺服器上進行資料更新。

附錄的 Figure 2 說明了當前 NuJudger 預期的微服務架構。

透過新增從服務數量，我們可以盡可能得減少評測排隊的問題，在遇到評測提交高峰時能夠盡可能處理評測要求，使評測佇列發生凍結的時機大幅減少。

(四) 軟體穩定與文件化

評測機作為程式評測系統最重要的核心，我們必須確保評測機的穩定性，NuJudger 具有以下的軟體穩定策略：

- 具有 95% 以上的測試覆蓋率，透過 Codecov 監督每筆提交是否滿足測試覆蓋率門檻
- 使用 GitHub Actions 持續整合，用於確認以下事項是否滿足：
  - 測試 Docker 是否能夠佈署。
  - 每筆提交皆測試軟體是否通過測試。
  - 靜態型別檢查。

除此之外 NuJudger 也具有文件化措施，例如使用 Swagger 制定 API 規範，利於使用 PDD 的方式進行開發、使用 ReadTheDocs 來製作使用指南，利於使用者架設服務。

五、 結論

透過 NuJudger，能夠使當前的評測服務更加穩定且安全，我們期望所有的 Online Judge 開發者使用 NuJudger，能夠確保服務穩定、安全且保證了使用者的體驗。除了沙盒服務能夠隔絕程式運行環境、使程式運行具有權限限制以及資源使用限制之外，我們也透過了良好的開發過程，例如：Code Review、足夠數量與顆粒度足夠小的單元測試、靜態型別檢查以彌補 Python 動態型別導致軟體不穩定的問題，我們也大幅的保證了核心的軟體品質。

期待在未來的不久，教育單位或比賽單位可以不用額外考量資安風險而依然堅持全人工或半人工批改程式，以及學生能夠使用程式評測平台更快速、快樂的學習程式。

參考文獻

[1] M. Forišek. Security of programming contest systems. 2007.

[2] A. Kurnia, A. Lim, and B. Cheang. Online judge. *Computers & Education*, 36(4):299–315, 2001.

[3] M. A. Revilla, S. Manzoor, and R. Liu. Competitive learning in informatics: The uva online judge experience. *Olympiads in Informatics*, 2(10):131–148, 2008.

[4] S. Wasik, M. Antczak, J. Badura, A. Laskowski, and T. Sternal. A survey on online judge systems and their applications. *ACM Comput. Surv.*, 51(1), jan 2018.

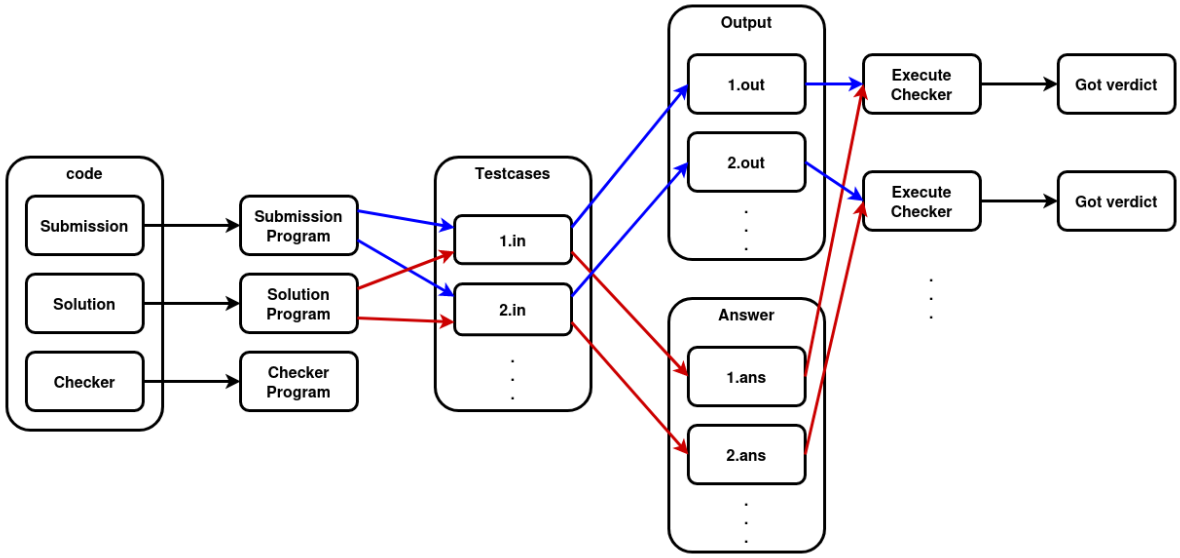

Figure 1: NuJudger 評測機制

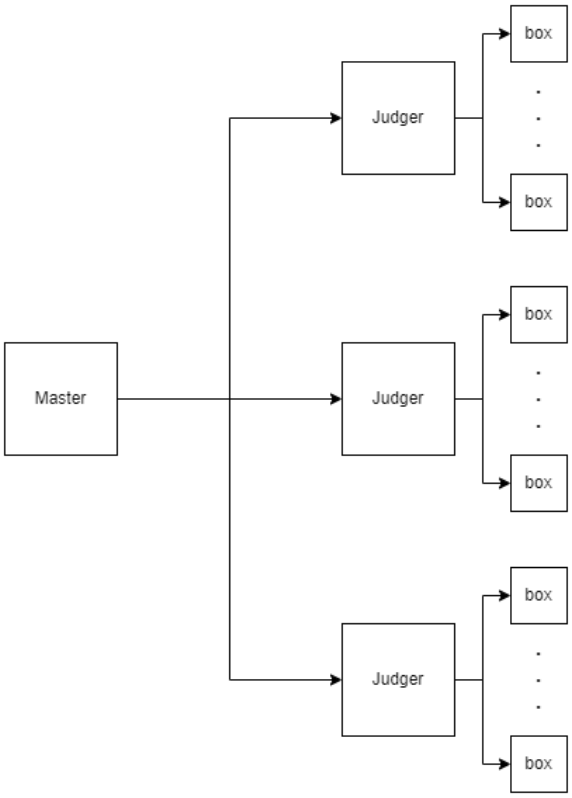

Figure 2: NuJudger 微服務架構