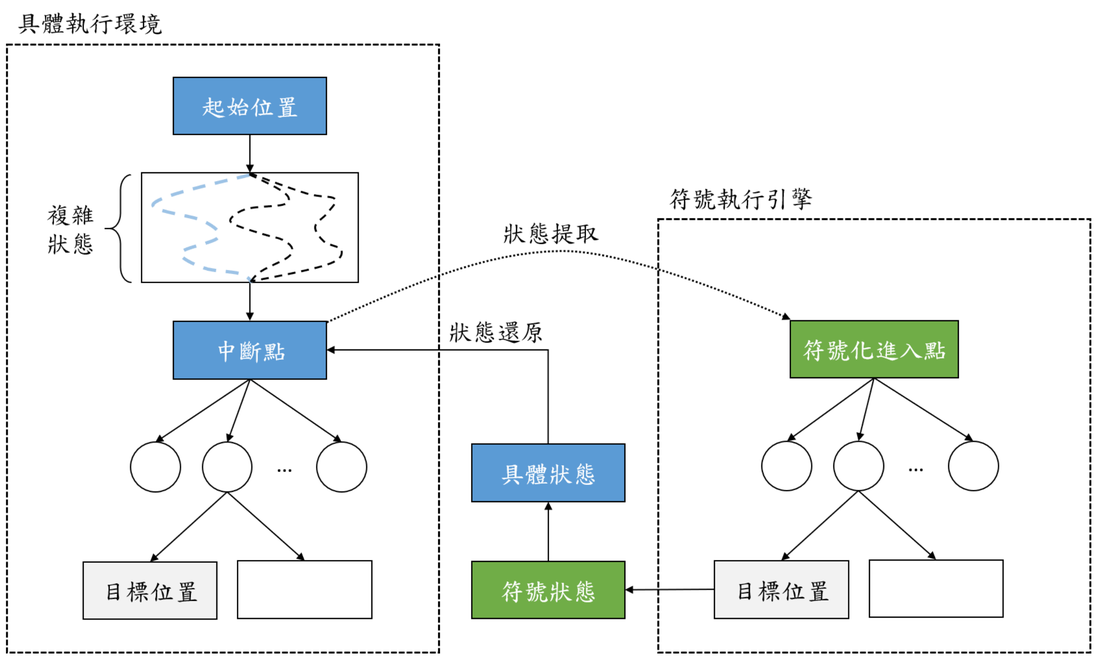
程序運行期間之原位符號執行及其應用
In-Situ Symbolic Execution for Running Processes and Its Application

研究生:呂翰漳　　指導教授：林一平教授、吳育松教授
國立陽明交通大學　　資電亥客與安全碩士學位學程

1. 摘要

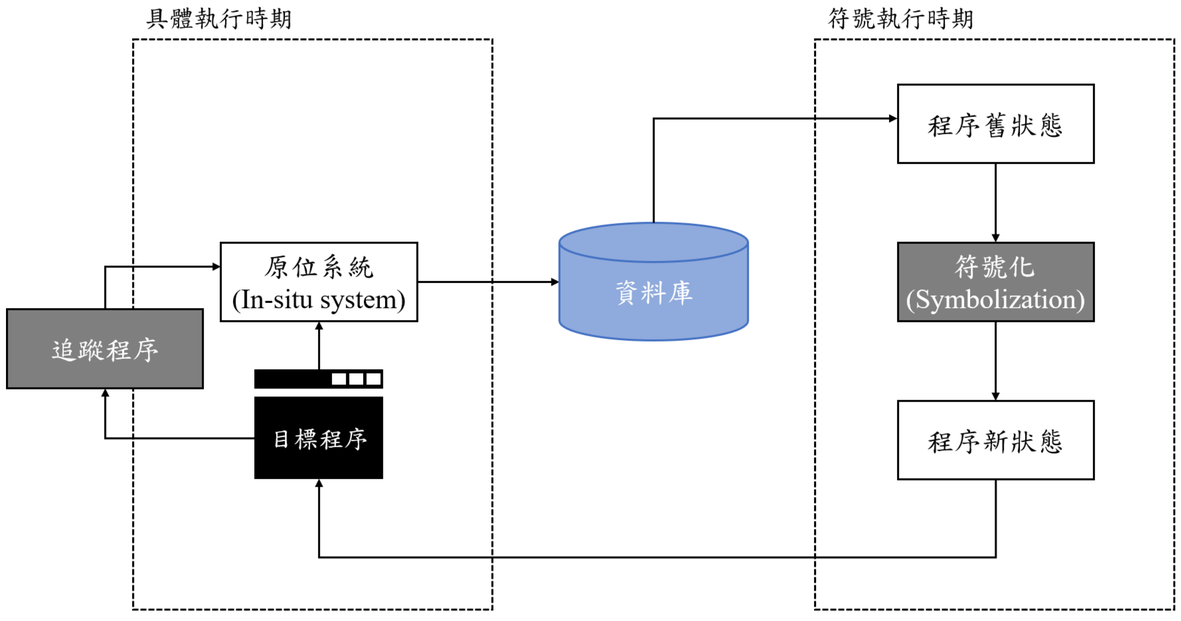
近年來資安漏洞層出不窮，為了找到能夠觸發這些漏洞的條件，我們必須提高程式碼覆蓋率，符號執行即為一種。不過符號執行仍然存在許多的限制，使得多數大型系統無法充分利用其功效。如果大量的使用第三方函數庫或與系統呼叫，則容易使符號執行引擎產生不穩定的因素，造成錯誤的執行狀態。且不穩定的狀態容易使分析系統的複雜度增加，往往需要耗費大量的分析時間。而許多程序有漏洞疑慮的範圍通常不大，基於以上原因，在本研究中我們提出一種新的技術，稱為原位符號執行 (In-situ Symbolic Execution)；不須將完整的程序作用於符號執行引擎，僅在特定區段使用符號執行，再將執行結果同步回原程序中。使目標程序可以有效的跳過其中複雜的部份，並讓其在具體環境中得以繼續執行。我們將選擇不會進入第三方函數庫的程式區段作分析，最終證實不僅可以使得程序在具體環境與符號環境交互執行，且相較於傳統符號執行有更好的執行效能。

2. 系統設計



原位符號執行指的是程序正常執行期間在原指令執行位址使用符號化執行，等待符號執行完後再返回程序正常執行，如上圖所示。具體執行環境可於程序隨意位置直行至欲符號化的位置開始使用符號化引擎，開始使用符號執行直到運行至目標位置，將符號狀態求解成具體狀態後導回具體執行環境的欲符號化的位置，如此即可使用具體執行環境直行至目標位置。使用原位符號執行的做法能達成以下三點：第一點是可以在任意程序中任意位置使用符號化執行；第二點是不須理會原指令位置前複雜的符號狀態，因為指令前的狀態皆為程序執行期間的具體狀態；第三點是可以使用正常具體執行的方式，而不必運行在符號執行的引擎上，意味著程序可以使用一般方式運作，在需要符號執行時再轉換即可。

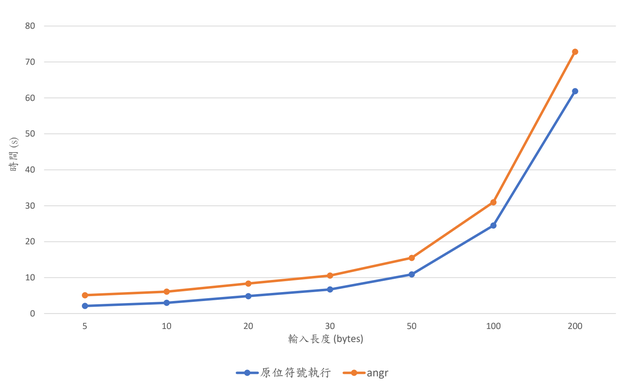
本系統主要分為兩大部份：第一部份為原位系統，此處的行為模式為具體執行，主要目的為暫停與繼續執行目標程序，以及將所需的程式碼和記憶體資訊切割下來交給符號執行引擎使用；第二部份為符號執行引擎，此處為進行符號執行的區域，將變數或是記憶體內容符號化使其能夠運行至目標位置，再求得符號化內容，最後將結果同步回目標系統，讓系統可以順利運行。



原位系統主要功能為暫停目標程序及擷取所求的片段程式碼，再記錄系統當下的狀態，使用這些數據建構出可將片段程式符號化的效果，等符號執行完畢後才同步回具體執行之中。在目標系統運行以前，我們會先利用逆向分析的方式找出此系統需要被符號執行的區塊以及欲執行到的目標位置，而這些區塊須由一個至數個基本塊所組成。當系統執行到此一區塊時會將此程序暫停，交由符號執行引擎去運作至所設定的目標位置，最終將得到的結果返回至目標程序，使其可以順利執行到目標位置。執行層面分為運行時程序狀態的擷取其內容包括程式碼的與記憶體內容的擷取、符號引擎的執行包含了同步至符號執行引擎以及符號化執行、狀態還原三大步驟。

3. 實驗與結果

本研究分別針對了驗證符號執行的正確性、符號執行在真實程式的適用性、應用實例進行實驗。其中符號執行的正確性中測試了複雜運算及資料長度與時間的關係；真實程式舉了bash5.0及gzip1.10等已發佈的系統中的函數作為實驗對象；而應用實例則列舉了條件是汙染分析及反制動作生成作為實驗資料集。



分析對象 / bash5.0函數	總時間	記憶體設置時間	符號執行時間	指令數量
set_shell_name()	1.98 s	0.36 s	0.45 s	76
array_shift()	3.31 s	0.30 s	1.41 s	207

分析對象 / gzip1.10函數	總時間	記憶體設置時間	符號執行時間	指令數量
huft_build()	4.05 s	0.40 s	1.98 s	298
built_tree()	5.51 s	0.31 s	3.43 s	154

分析對象 / 變數長度	分支數	可到目標位置的分支數	汙染5 btyes的分支數	汙染9 btyes的分支數	符號執行時間
1 byte	256	37	36	1	16.50 s
2 bytes	12148	847	792	55	511 s

4. 結論

在本研究中我們提出一套使目標程序在具體執行期間使用符號化引擎執行，結束時再同步回原目標程序的新系統，稱為原位符號執行。本系統不僅使符號執行得以更輕量化的執行，還避免了環境交互作用所帶來的副作用；且得益於此系統交互執行的原理，可以使目標程序完美的繼續執行。我們於實驗章節驗證原位符號執行系統的可行性與相對於傳統符號執行所帶來的高效能；並證實在bash5.0與gzip1.10的部份函數中可以順利運作本系統且正確還原至具體環境；最後在應用實例的部份將模擬出的可用場景一一實現，包括污染追蹤與反制動作。目前尚未處理動態連結函數庫的程式碼片段，因為地址空間過大造成擷取部分尚無法完善。