國立交通大學

資訊科學與工程研究所

碩士論文

雲端軟體弱點探索分析資料庫

A Cloud-based Benchmark Database for Software Vulnerability Analysis and Discovery

研 究 生：趙正宇

指導教授：黃世昆　教授

中 華 民 國 一 百 零 二 年 六 月

雲端軟體弱點探索分析資料庫

A Cloud-based Benchmark Database for Software Vulnerability Analysis and Discovery

研 究 生：趙正宇　　Student：Chen-Yu Chao

指導教授：黃世昆　　Advisor：Dr. Shih-Kun Huang

國 立 交 通 大 學
資訊科學與工程研究所
碩 士 論 文

A Thesis
Submitted to College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
In

Computer Science

June 2013

Hsinchu, Taiwan, Republic of China

中華民國一百零二年六月

雲端軟體弱點探索分析資料庫

學生：趙正宇　　指導教授：黃世昆

國立交通大學資訊科學與工程研究所碩士班

摘要

觀察過去 Stuxnet、APT 與最近 Google、Facebook 和微軟遭攻擊事件，網路世界的戰爭已不容輕視，其中所使用的武器便是針對各種軟體弱點的攻擊程式。本研究擬基於雲端系統建置一個軟體弱點探索分析資料庫，儲存可執行的軟體弱點環境，同時也改善實驗室開發的自動脅迫產生器(Automatic Exploit Generator, CRAX)，與此資料庫整合，利用雲端系統自動化軟體弱點的探索過程，除能針對軟體弱點自動產生脅迫(Exploit)外，還可將建置的實驗環境轉換為 wargame，提供人員安全意識訓練的教材。

A Cloud-based Benchmark Database for Software Vulnerability Analysis and Discovery

Student: Cheng-Yu Chao Advisor：Dr. Shih-Kun Huang

Department of Computer Science and Engineering

National Chiao Tung University

ABSTRACT

Recent attacks like Stuxnet, APT, and on large corporations including Google, Facebook and Microsoft have caused much damage on valuable information asset. The Internet warfare can no longer be ignored.

In this thesis we developed a cloud-based benchmark database for software vulnerability analysis and discovery. This system is capable of maintaining executable environment of various software vulnerabilities. We integrate the automated exploit generation system (called CRAX) formerly developed by our laboratory into the system, taking advantage of cloud system to automate the software exploit writing process. The system not only provides the automatic exploit of software vulnerability but can also construct a wargame for training security expertise from emulated environment.

誌謝

感謝我的指導教授黃世昆老師，在大二時開了程式安全的課程，啟發了我對研究資訊安全的熱誠，指引學習方向，鼓勵我參加各種比賽。更要感謝老師對我平時生活上的照顧與關心、在我挫折時給的鼓勵、用激將的方式來刺激我向上進步的決心。

第二個要感謝的是大一大二時的導師王國禎教授，若沒有王老師當時替我頂住家裡的壓力，我可能就在第二次被二一時就要休學重考，更不要談能繼續學習資訊安全的技術，攻讀碩士學位。

再來要感謝我參加各種比賽的搭當，蘇宏麒、許晏俊、劉歡、陳俊諺，一起在比賽中研究各種攻擊手法，學習新知識，奠定了這個論文的基礎。

感謝口委對我論文的各種建議，及在我文筆不好的情況下，仔細地閱讀並修改其中不通順的地方，還有口委在聽完簡報後給的認可。

最後感謝我的家人與女友，在完成學業的過程中，對我的包容與鼓勵。尤其感謝女友對我論文的協助，由於本身很不會寫文章，若沒有他平時強迫我討論文藝的話題，仔細地幫我修正寫出來的字句，寫出來的文字可能會更糟。

目錄

表目錄

圖目錄

第一章 緒論

1-1 研究動機

隨著資訊的發展，網路已是當今社會不可或缺的溝通管道，但過度依賴網路，也讓駭客攻擊造成的危害越來越大。隨著各種行動裝置興起，實體隔離的網路環境也飽受威脅。典型的案例就是 Stuxnet[1]，攻擊者使用數個針對瀏覽器和 Windows 作業系統的零日(Zero Day)攻擊，透過網路散佈病毒，被感染的電腦會將針對核子設施病毒植入連接此電腦之 USB 隨身碟中，最後已遭植入病毒的隨身碟被帶至隔離網路，感染隔離網路內的電腦，進一步尋找工業 PLC 控制器再加以破壞。

以上所提及的零日攻擊是指一個軟體或作業系統被發現漏洞，且此漏洞可能造成伺服器癱瘓，或攻擊者可以直接執行任意程式碼，只要在此軟體漏洞被公佈前所作的攻擊，即可稱為零日攻擊。零日攻擊在被公佈前沒有人可以預防，防火牆、防毒軟體等都無法偵測。

防止零日攻擊的方式，除了加強程式設計師的資訊安全觀念，避免程式有漏洞讓駭客有機可趁之外，研究出有效率的軟體安全測試及檢查方法是大勢所趨。

由於目前缺乏弱點資料庫的評比資料與執行環境，現有的弱點資料庫皆為靜態不可執行的資訊，造成過去的弱點程式版本與執行環境取得不易，所以我們希望建立一個可重複驗證的弱點資訊資料庫，並藉此進行弱點檢查方法的研究。

1-2 研究目標

本研究擬設計基於雲端運算的軟體安全性測試系統，使用標準化的方法測試不同作業系統、應用程式和網頁的安全性，冀望此系統能夠快速大量地進行軟體安全性測試，驗證弱點可能造成的危害程度，並建置出一個可執行、測試的軟體漏洞資料庫。

最後這些資料，都可以轉換為 wargame，訓練程式開發者提升安全程式開發能力，自行體驗平時為趕時程而不嚴謹的程式撰寫方式，所可能造成的嚴重安全性問題。

1-3 主要貢獻

提供使用者友善的介面，一般使用者即可自行測試軟體安全程度，並將過去曾發生的軟體漏洞設計成 wargame，提供實際體驗的資訊安全教育訓練材料。

此外，CVE 僅提供漏洞資訊，Exploit DB 僅提供攻擊程式，有弱點的軟體在資訊安全研究中是珍貴且難以取得的資料。本系統除了測試之外，也完整保留弱點程式與執行環境，以提供日後研究所需。

第二章 背景

2-1 符號執行

這是用來動態分析程式執行路徑的方法[2, 3]，以符號變數取代一般輸入資料。符號變數並非固定值，而是根據程式的執行狀況動態改變。當符號變數影響到其他變數時，被影響的變數也會被設成符號變數，相對應的關係式也會被記錄。

符號執行遇到分支條件(branch condition)，若有符號變數參與，就會產生兩個狀態對應此分支條件的兩種結果，並且加入分別對應這兩種不同路徑的限制式(path constraint)，當符號執行再次遇到分支條件時，產生的路徑限制式會與原本的路徑限制式合併。若新產生的路徑限制式若與先前產生的路徑限制式衝突，則直接放棄，代表此路徑已經執行完畢。

完成所有路徑探索後，產生的每一個路徑限制式代表著程式不同的執行路徑，再透過解譯器(solver)，產生可滿足路徑限制式的測資，限制其在程式執行時有相同的路徑。

```
01.  foo(int x, int y){
02.      if( x > y ){
03.          x = x - y;
04.          if( x < 0 )
05.              return false;
06.      }
07.      print x , y
08.  }
```

圖 2-1 Symbolic Execution Example

舉例來說，此段程式以符號執行方式，將傳入的參數設定成符號變數，當程式執行到第一個分支點(Branch)(第二行)的判斷式時，會分成兩條路徑執行 x＞y 和 x≦y。x＞y 這條路徑會進入第三行，此時符號變數互相影響，

紀錄影響的關係式，接著進入第四行第二個分支點 x＜0，由於此 x 已被紀錄成 x-y 故判斷式會被轉換成 x－y＜0，此結果與前面產生的路徑限制式(x＞y)衝突，路徑便被捨棄，結束此路徑的探索，僅剩下負向結果 x－y≦0。而第一個分支點的另一條路徑 x≦y 後續並無其他分支點，故也完成探索，最後產生兩組路徑限制式，對應兩條程式執行路徑，如圖 2-2。

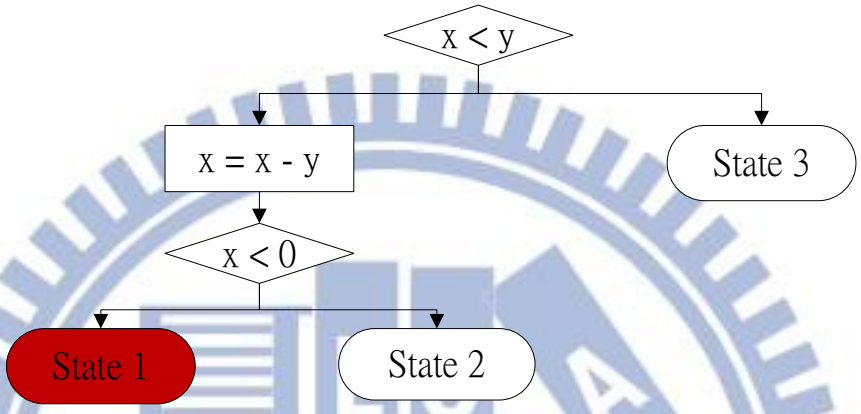


圖 2-2 Symbolic Execution Example Flowchart

由於符號執行會有路徑數量爆炸的問題，程式分支數量會導致執行路徑數量呈指數性增加，遇到無窮迴圈時，會直接產生過多的路徑。故符號執行必須給定一個執行範圍，限定迴圈上限，或有路徑選擇策略，擬真執行(Concolic Execution)[4]便提供了一種路徑選擇的方法。

擬真執行

擬真執行與符號執行最大的差別在於符號執行沒有初始的測試資料，遇到分支條件時，才針對符號變數產生符合分支條件的路徑限制式，而擬真執行則是先有一組測試資料，使用此測試資料產生執行路徑的限制式，再藉由否定(negate)所遇到的分支限制式(branch constraint)產生新的程式執行路徑，如圖 2-3。

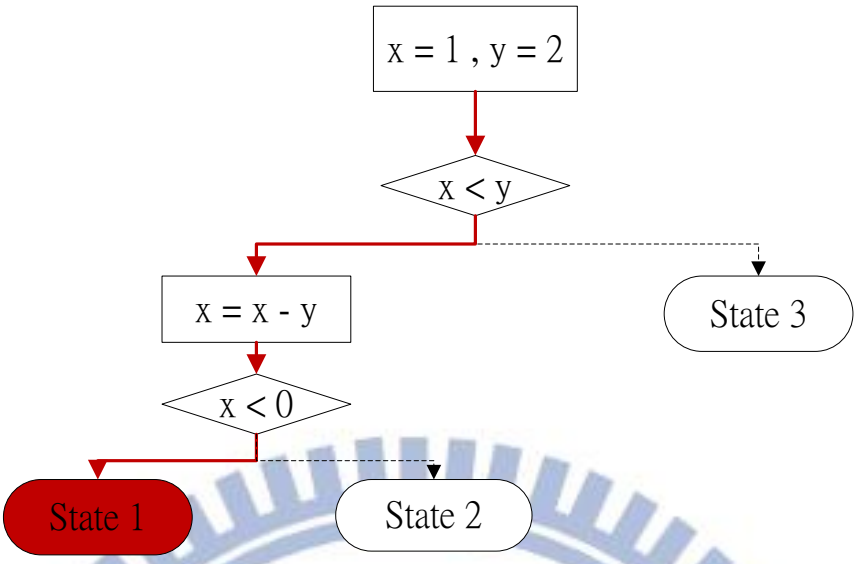


圖 2-3 Concolic Execution Example Flowchart

KLEE

KLEE[5]是基於 LLVM[6]所建立的虛擬符號執行機器，也最先提出符號環境的概念：將相關輸入的資料流轉換為符號輸入來測試執行檔。較常見且在 CRAX 中有實際利用的符號環境有 Symbolic Standard Input、Symbolic Socket、Symbolic File、Symbolic Environment Variable、Symbolic Argument Variable。

S^2E

S^2E[7]結合 KLEE 與 QEMU[8]，提供一個完整且有效率的符號執行環境，支援一般模式(User mode)與核心模式(Kernel mode)的符號執行，且能在符號執行和擬真執行兩者中做切換，跳過不相關的地方或是使用擬真執行，來加快符號執行的速度。

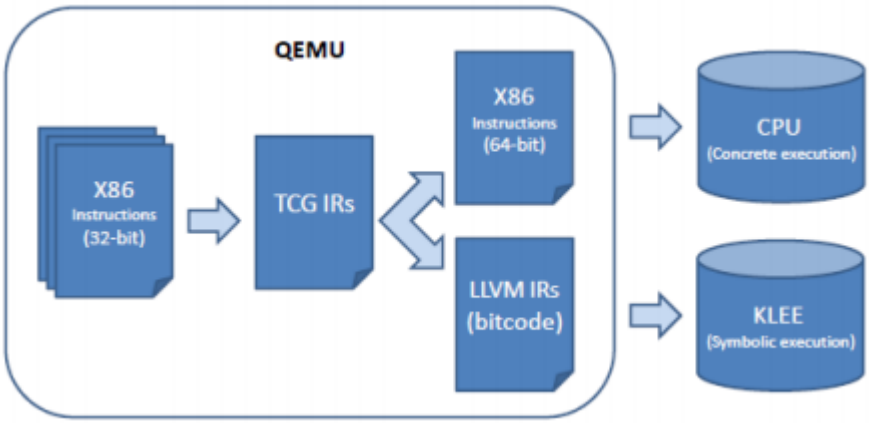


圖 2-4 S^2E 架構

2-2 相關工具

Qemu

可模擬各種指令架構的虛擬機器，具有高效率及跨平台的優點。我們研究中使用 Qemu 建立各種作業系統的映像檔，並運用映像快照功能，來對已安裝弱點軟體的虛擬機器產生快照。

Libvirt

提供標準化的虛擬機器管理介面，支援各種 VMM (Xen, KVM, Qemu, VMware, Virtual Box 等)，可輕鬆管理虛擬機器、虛擬網路、及儲存設備。

Cloud-init

Ubuntu 處理雲端客製化的一個套件，可在映像檔初次啟動時，自動執行預先設定之任務，例如設置 IP、主機名稱、插入外部檔案，安裝其他套件等工作。此套件提供的功能讓我們能用簡單的腳本程式、少量的設定檔，代替虛擬機器的手動建置與部署。

2-3 相關研究

2-3-1 弱點資料庫

CRAX (Crash Analysis and Automatic Exploit Generation [9])

目標擬自動化偵測程式漏洞並產生可執行任意程式碼的攻擊，在駭客造成嚴重損失前，發現漏洞。只要給予能造成受測程式不正常運作的測試資料，CRAX 就利用此測試資料，以符號執行檢查程式執行路徑，並嘗試產生脅迫(exploit)。

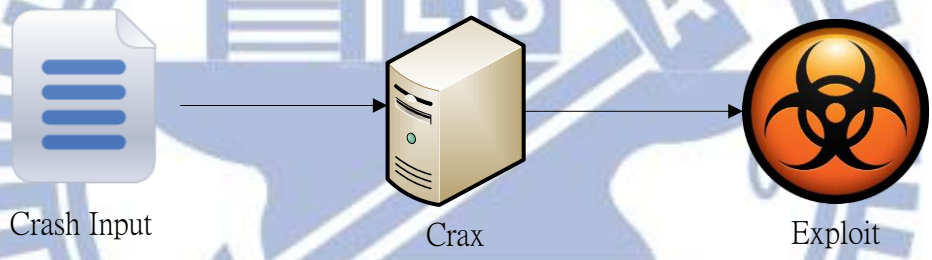


圖 2-5 Crash Analysis and Automatic Exploit Generation

CRAX 執行前，先設定受測程式，使用 S^2E 所修改的 Qemu 進行環境建置。根據受測程式的類型(symfile , symsocket , symenv 等)，植入相對應的測試腳本，在啟動實驗前，使用 Qemu 快照功能儲存機器狀態，以帶有符號執行功能的 Qemu 回復快照，才能正式開始符號執行並進一步產生脅迫。在這些步驟中，必須頻繁的在主機端及虛擬端來回切換操作，因此每次實驗資料的修改，都必須花費很多的時間處理這些映像檔的快照。

CVE

CVE (Common Vulnerabilities & Exposure [10])規範每個被曝露出的資訊安全漏洞，制定唯一編號並提供軟體版本、問題描述、危害等級、漏洞修補等資訊。作為研究軟體弱點時共同的參考，可快速交換軟體弱點之詳細描述與規格。

例如 CVE-2013-0422 是危害等級 10 的漏洞， 影響範圍在所有 Java 7 update 11 之前的版本，可讓遠端駭客透過 Java Reflection API 繞過安全檢查，並執行任意程式碼。

Vulnerability Summary for CVE-2013-0422

Original release date: 01/10/2013

Last revised: 02/26/2013

Source: US-CERT/NIST

Overview

Multiple vulnerabilities in Oracle Java 7 before Update 11 allow remote attackers to execute arbitrary code by (1) using the public getMBeanInstantiator method in the JmxMBeanServer class to obtain a reference to a private MBeanInstantiator object, then retrieving arbitrary Class references using the findClass method, and (2) using the Reflection API with recursion in a way that bypasses a security check by the java.lang.invoke.MethodHandles.Lookup.checkSecurityManager method due to the inability of the sun.reflect.Reflection.getCallerClass method to skip frames related to the new reflection API, as exploited in the wild in January 2013, as demonstrated by Blackhole and Nuclear Pack, and a different vulnerability than CVE-2012-4681 and CVE-2012-3174. NOTE: some parties have mapped the recursive Reflection API issue to CVE-2012-3174, but CVE-2012-3174 is for a different vulnerability whose details are not public as of 20130114. CVE-2013-0422 covers both the JMX/MBean and Reflection API issues. NOTE: it was originally reported that Java 6 was also vulnerable, but the reporter has retracted this claim, stating that Java 6 is not exploitable because the relevant code is called in a way that does not bypass security checks. NOTE: as of 20130114, a reliable third party has claimed that the findClass/MBeanInstantiator vector was not fixed in Oracle Java 7 Update 11. If there is still a vulnerable condition, then a separate CVE identifier might be created for the unfixed issue.

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 10.0 (HIGH) (AV:N/AC:L/Au:N/C:C/I:C/A:C) (legend)

Impact Subscore: 10.0

Exploitability Subscore: 10.0

圖 2-6 CVE-2013-0422

Exploit DB

Exploit Database[11]為一種整理各種弱點利用程式的資料庫，類似於CVE，對每一個漏洞利用程式制定編號。兩者最大的不同，在於前者提供各種漏洞利用的驗證程式(Proof Of Concept)或直接針對該漏洞的脅迫(Exploit)，少部分則提供有問題的軟體版本。

Metasploit

這是一種滲透安全測試工具[12]，提供軟體弱點滲透測試和 IDS 特徵研究所使用的工具，支援各種弱點利用的攻擊程式。不同於 Exploit DB，此工具提供的攻擊程式具有各種調整選項，根據使用者需求客製化攻擊，自動置換 shellcode，並有容易操作的使用者介面。

除上述功能外，Metasploit 亦支援 Opcode Database，讓安全研究人員可遵循其規範，能在 Metasploit 上加入自己所撰寫的攻擊程式。

2-3-2 實驗安全平台

Wargame

由於直接對主機進行攻擊是違法的，在需要學習資訊安全技術時，就只能自己架設虛擬環境來練習。例如 Webgoat 就是這種模式的代表，但自行架設虛擬環境不夠真實，也比較沒有成就感，因此慢慢就演化出了各種模式的 wargame，提供合法的駭客攻防練習環境。

圖 2-7 NCTU Online Judge

NCTU Online Judge[13]是一個能夠讓使用者對各種系統、軟體上的漏洞進行攻擊的系統。利用這個平台，使用者能模擬各種利用程式漏洞所進行的攻擊，充實使用者對網路攻擊的知識與經驗。

這個平台分成兩大部分，網頁端及系統端，網頁端為 apache + MySQL + LDAP 構成，提供使用者介面。系統端則是由一台提供給使用者的工作站及一台負責開啟漏洞程式的題目伺服器。題目伺服器為了將每個使用者開啟的漏洞程式隔離，使用 FreeBSD 的 chroot，但也因此造成了題目增加不易，目標程式必須相容於 FreeBSD 等問題。

Testbed@TWISC

成功大學基於美國 Utah 大學所授權的 Emulab 軟體所建立的網路安全測試平台[14]，用一整樓的機房、實體伺服器，提供資訊安全研究所需要的各種網路、系統環境。

CSEP

中央大學雲端安全實驗平台[15]，類似 NCTU Online Judge，提供預先設置好的環境供使用者體驗攻擊與防禦。

第三章　研究方法

前一章所剛提到的弱點資料庫，未提供有弱點的軟體版本，也沒有弱點軟體的可執行環境。而國內現有的測試平台，也沒有這些弱點軟體的實驗環境。

因此，我們希望自動化 exploit 產生過程(CRAX)，建置一個基於雲端技術的軟體弱點資料庫，自動尋找軟體弱點並驗證此弱點的影響程度，同時快速產生各種含有弱點軟體的執行環境。運用雲端技術搭配前述所建置的環境亦可解決 wargame 所面臨的問題。

3-1　AutoCRAX

為了自動化 CRAX，我們將 CRAX 的實驗分為六個階段：建立映像檔快照、產生測資、設定實驗、運算節點控制、產生脅迫及結果紀錄。

其中建立映像檔快照、產生測資都是實驗的前置準備，搭配後台管理來迅速建立所需要的實驗環境。接著在設定實驗階段新增實驗，AutoCRAX 便會自動新增運算節點，運算節點從實驗隊列裡取出實驗資料開始產生脅迫，最終將實驗記錄傳回中央控制端分析結果。

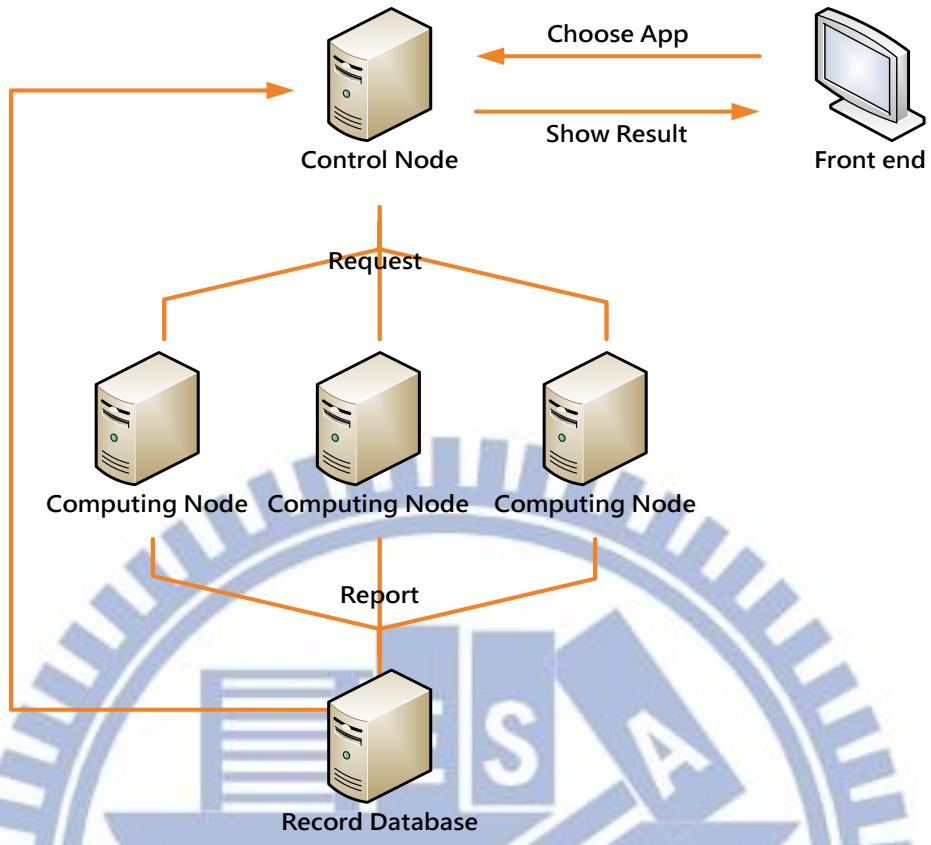


圖 3-1 AutoCRAX 流程圖

3-1-1 建立映像檔快照

原本在建立 CRAX 實驗所需的映像檔快照時，必須頻繁的在主機端與虛擬端間切換，而每更動實驗參數就必須重新建立一次映像檔快照，為了避免不必要的操作，我們使用類似 Cloud-init 的機制，在安裝受測程式後，即植入控制程式以便從外部控制受測環境。

我們簡化建置環境的過程，盡可能減少人工操作，將主機端所需要做的動作交給 Libvirt 在背景操控。客戶端的部分則是利用網頁介面直接操作，並直接將所安裝的程式版本資訊、映像檔名稱、快照編號相關資訊匯入資料庫。

3-1-2 產生測資

而我們所需要的測資，在 CRAXWeb 可以直接使用網頁爬蟲工具加上自己寫的分析程式把輸入的變數分離出來。CRAXUnix 的部分就需要未來預計完成的 CRAXFUzz 自動產生側資，或是使用已知會造成程式崩潰的輸入資料。

3-1-3 設定實驗

然後我們為 CRAX 實驗做一個簡單的 UI 來控制新增實驗，將實驗相關資料像是目標程式、輸入的側資、希望產生的攻擊等全部透過網頁設定，將實驗加入到實驗隊列裡面，並監控實驗隊列的情況，若太多則自動開啟額外的運算節點處理實驗。

3-1-4 運算節點控制

根據實驗隊列的大小，我們使用 Libvirt 控制 hypervisor 動態增加或減少運算節點，每個節點開啟時根據 CRAX 類型(CRAXWeb[16]、CRAXUnix[17]、

CRAXDroid[18]等)作相對應的初始化動作(CRAX-Init)，自動進入產生脅迫階段，由中央控制端監控所有節點的處理狀況。

3-1-5 產生脅迫

我們將 CRAX 產生脅迫的過程簡化成三個步驟，主機端請求、虛擬端請求、主機端回報。

當一個運算節點準備開始產生脅迫時，便發出主機端請求，從中央控制端取得實驗資訊（映像檔、目標程式、快照編號、shellcode 等），根據取得的資料設定 S^2E 並開啟目標映像檔快照。

開啟後的執行環境會發出虛擬端請求，從中央控制端取得測試資料、環境參數設定、程式啟動指令後，開始符號執行，嘗試產生脅迫。符號執行結束後發出關閉訊號至主機端。主機端收到關閉訊號便將本次實驗所產生的紀錄檔、脅迫等資料送回資料庫，供日後的分析使用。

3-1-6 紀錄分析

中央控制端除了儲存所有實驗紀錄外，根據實驗類型分析結果，檢查 EIP 是否被汙染，是否可以產生脅迫等資訊存入資料庫，若成功產生脅迫時，則開啟額外的運算節點用 NCTU Online Judge 驗證使用者過關的方式確認脅迫是否正確。

這些實驗結果統計分析和執行環境，就是我們資料庫所需要的東西。

3-2 雲端化 Wargame

原始 wargame 是將有問題的應用程式伺服器用使用者的編號開啟，再根據使用者的編號和題目組合產生唯一的過關金鑰，但在這種機制下，能做成題目的應用程式不多，大部分都必須修改原程式或是特製化設定檔，針對應用程式的需要去打造 chroot 所使用的環境。

我們將 AutoCRAX 運算節點控制的機制導入到 wargame，替代原本 chroot 的方法，當使用者開啟一個題目時，直接創建一個運算節點，透過 CRAX-Init，植入漏洞程式、使用者通關金鑰、為每種攻擊模式特製化設定系統。由於每個開啟的題目都是獨立的運算節點，因此可以提供涵蓋客戶端攻擊、阻斷服務攻擊、網頁攻擊和針對作業系統核心類型的攻擊模擬題目。

3-2-1 網頁攻擊

目前最常見的是針對網頁應用程式的攻擊，因為全球資訊網(World Wide Web) 的方便性，資訊取得容易，不需要太多的前置準備去尋找系統資訊及網路結構，且漏洞的利用相對其他攻擊來說也較簡單。開放網頁軟體安全計畫(OWASP)長期致力於協助改善網頁應用程式的安全性，每年都會公布年度前十大網頁弱點，也設計了針對網頁攻擊練習的 Webgoat 練習平台，但 Webgoat 練習平台是在自己的電腦上安裝，環境不夠真實、較沒挑戰性是他的缺點。

原本的 chroot 架構無法模擬網頁應用程式的 wargame，如今只需將 CRAXWeb 建置完成的實驗環境稍作修正，即可直接轉換為 wargame，可模擬對真實伺服器的攻擊，比 Webgoat 更加擬真。

3-2-2 作業系統核心

有些攻擊直接針對作業系統核心，可以直接取得最高權限，或是從遠端攻擊取得系統權限，舉例而言：CVE-2009-1041 是一個 FreeBSD 核心整數參數溢位的漏洞，透過此漏洞可以修改作業系統核心的記憶體資料，進一步取得系統管理者權限。

這種類型的攻擊因為牽涉到作業系統的核心，在原本 chroot 環境下，可能造成環境隔離失效，甚至造成系統崩潰，並且最大的問題是無法模擬各種作業系統版本。透過運算節點控制，我們可以開啟各種作業系統版本的運算節點當作使用者的攻擊目標，讓使用者嘗試作業系統核心的攻擊，避免攻擊失敗時系統崩潰，而影響其他使用者。

3-2-3 客戶端攻擊

與一般攻擊的題目不一樣，此類型的攻擊是由攻擊者事先架設伺服器，等待受害者連線至伺服器後，觸發受害者機器上客戶端程式的漏洞，最典型的例子就是跨站腳本攻擊(Cross Site Script)，在網頁伺服器上植入惡意程式碼，攻擊連線到該網頁伺服器的客戶端主機。

我們只需要在開啟運算節點，植入有漏洞的程式後，再使用該程式連線使用者指定的伺服器，就能模擬各種類型的客戶端攻擊，包含但不限於跨站腳本攻擊。

3-2-4 阻斷服務攻擊

這裡的阻斷服務攻擊指的並不是分散式阻斷服務攻擊，而是針對應用程式的漏洞做出特製化的輸入資料，讓應用程式崩潰無法正常工作的攻擊。

針對這種攻擊的模擬，不能使用原本的金鑰驗證方式，而是在控制端直接檢查開啟的運算節點是否正常工作，若沒有正常工作則判斷使用者攻擊成功。

3-2-5 分析與鑑定

透過雲端化的系統，可以嘗試分析鑑定類型的 wargame。以往此種類型的 wargame，提供側錄的封包檔案、系統紀錄、硬碟或記憶體狀態的映像檔，而現在可直接備份被入侵過的機器狀態轉換為 wargame。訓練使用者找出入侵的途徑、來源、補救方法等，讓使用者能體會真實的入侵分析與鑑定。

第四章　研究結果

4-1　標準化 CRAX 實驗流程

原本 CRAX 實驗為了因應各種型態的實驗，建立映像檔的規則、符號執行的方式、實驗步驟都不盡相同， 導致建立實驗非常沒有效率，每一個實驗必須至少花費三小時以上才能做好所需環境，最後經過各種數據測試才能完成一個最終版本的實驗環境，供日後重複實驗使用。

若要標準化 CRAX 實驗流程，除了加快建立實驗環境的過程外，更讓操作人員的學習曲線較為平緩，不需重新研究不同型態的 CRAX 實驗方法，只要掌握標準化的 CRAX 實驗流程即可。且標準化的實驗流程，將實驗環境的建立與實驗完全切割開，只要目標程式的實驗環境建立，就可以從網頁介面直接控制實驗變因，而不需要每改變一次測試資料就得開啟映像檔修改資料，重新做一個實驗環境的快照，節省時間如表 4-1。

	未標準化	標準化
建立環境	超過三小時	半小時以內
修正實驗	最少十分鐘	一分鐘以內
重現實驗	最少十分鐘	一分鐘以內

表 4-1 標準化 CRAX 實驗流程前後所需操作時間比較

目前已完成自動化 CRAXWeb 七個實驗，CRAXUnix 十四筆，如表 4-2、4-3 所列:

應用程式	連結數量	已完成實驗次數
eve	9	15
faqforge	7	12
schoolmate	269	4
Webchess	65	8
c_and_c_2.0.89	102	2
Testlink_1.8.4	218	2
Baviti	3605	0

表 4-2 CRAXWeb 已自動化實驗列表

應用程式	已建立測資	脅迫產生
aeon_0.2a	1	Complete
iwconfig_2.6	1	Complete
glftpd_1.24	1	Complete
ncompress_4.2.4	1	Complete
htget_0.93	2	Not Yet ...
Expect_5.43	1	Not Yet ...
rsync_2.5.7	1	Not Yet ...
acon_1.0.5	1	Not Yet ...
gif2png_2.5.3	1	Not Yet ...
hsolink_1.0.118	1	Complete
exim_4.41	1	Complete
aspell_0.50.5	1	Not Yet ...
xserver_0.1a	0	Not Yet ...
xmail_1.21	1	Not Yet ...

表 4-3 CRAXUnix 已自動化實驗列表

4-2 AutoCRAX 實驗控制端

AutoCRAX 實驗控制端提供友善的使用者網頁介面，直接操控完整實驗流程，從建立實驗開始，修正測資、運算節點監控到最後的完整實驗記錄統計，針對不同型態 CRAX 實驗的網頁服務。

4-2-1 建立實驗

透過 noVNC 直接從網頁將應用程式插入實驗映像檔，再根據不同型態實驗所需的參數，由控制端植入所需程式、映像檔快照與實驗環境的建立。CRAX-Init 實際介面如圖 4-1。

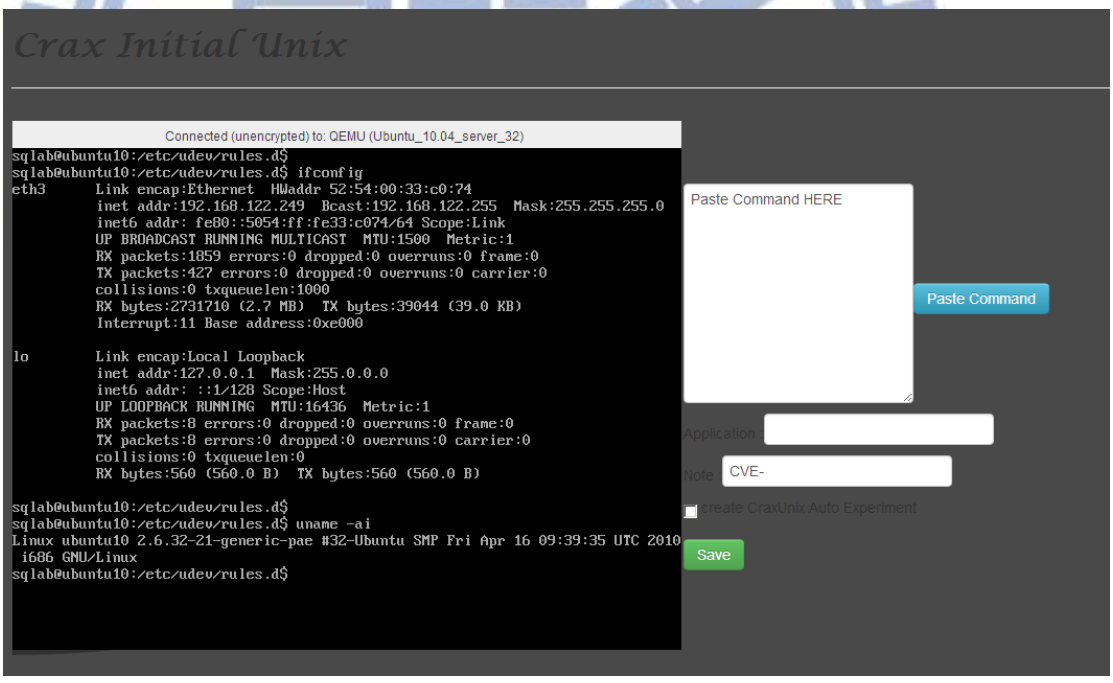


圖 4-1 建立實驗環境介面 CRAX-Init

透過 CRAX-Init 可以快速建立各種弱點軟體的可執行環境，表 4-4 列出了目前已建立完成之弱點軟體的可執行環境，除了 CRAX 實驗使用之外，也可以當成 wargame 題目，供練習資訊安全的攻擊與防禦。可改變原本 NCTU Online Judge 系統的限制，提供一個較簡單操作、成本較低、受測系統啟動時間較快的安全實驗平台，見表 4-5。

Application	OS	CVE
sudo-1.8.2	Debian_6.0.1_32	CVE-2012-0809
nginx-1.4.0	Debian_6.0.1_32	CVE-2013-2028
mplayer-1.0rc3-4.4.4	Debian_6.0.1_32	CVE-2010-3429
mysql-server-5.1.69	Ubuntu_10.04_server_32	None
apache-2.0.58	Debian_6.0.1_32	CVE-2006-3747
nginx-0.7.65	Ubuntu_10.04_server_32	CVE-2010-2263
lighttpd-1.4.26	Ubuntu_10.04_server_32	CVE-2013-1427
proftpd-1.3.2c	Ubuntu_10.04_server_32	CVE-2010-4221
koha-3.12-current	Debian_6.0.1_32	0day
PostgreSQL-8.4.17	Ubuntu_10.04_server_32	None

表 4-4 已建立之可執行軟體環境一覽

	維運費用	**新增環境**	**網路模擬**	**啟動時間**
Testbed@TWISC	極高	複雜	真實	較長
CSEP	低	複雜	部分	短
AutoCRAX	低	簡單	部分	極短

表 4-5 國內安全實驗平台比較

4-2-2 修正測資

在產生脅迫資料時，必須嘗試不同的輸入、環境參數設定或改變符號變數的長度。AutoCRAX 實驗控制端，針對每種型態的 CRAX 提供不同介面來改變上述的實驗變因。

以 CRAXUnix 的實驗為例，從網頁可直接修改輸入的測資、測資輸入方式(symfile、symstd 等)、欲植入的程式碼及一次符號執行最大可接受的負號變數長度。按下按鈕便會在實驗隊列裡新增一筆實驗，如圖 4-2。

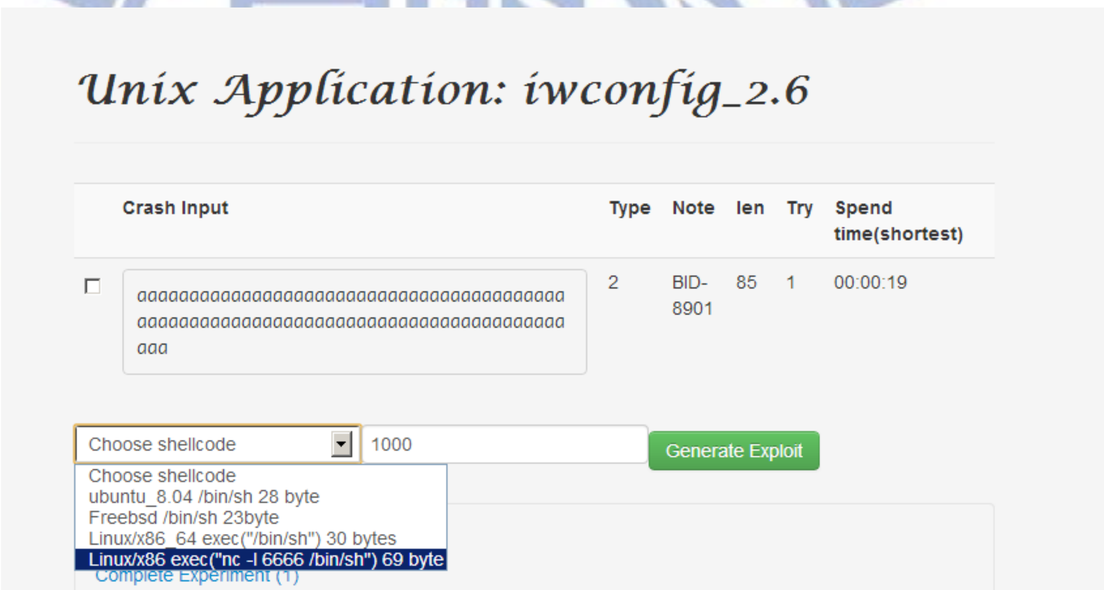


圖 4-2 修正測資控制介面

4-2-3 運算節點監控

AutoCRAX 控制端除了提供運算節點狀態清單之外，也提供即時的實驗進度監控及運算節點的主機端及客戶端的直接控制介面。如圖所示。

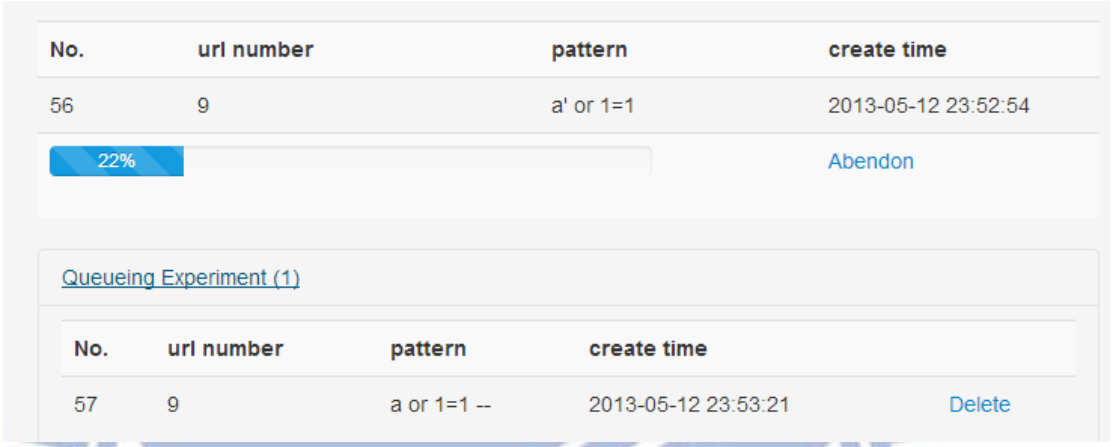


圖 4-3 實驗進度監控

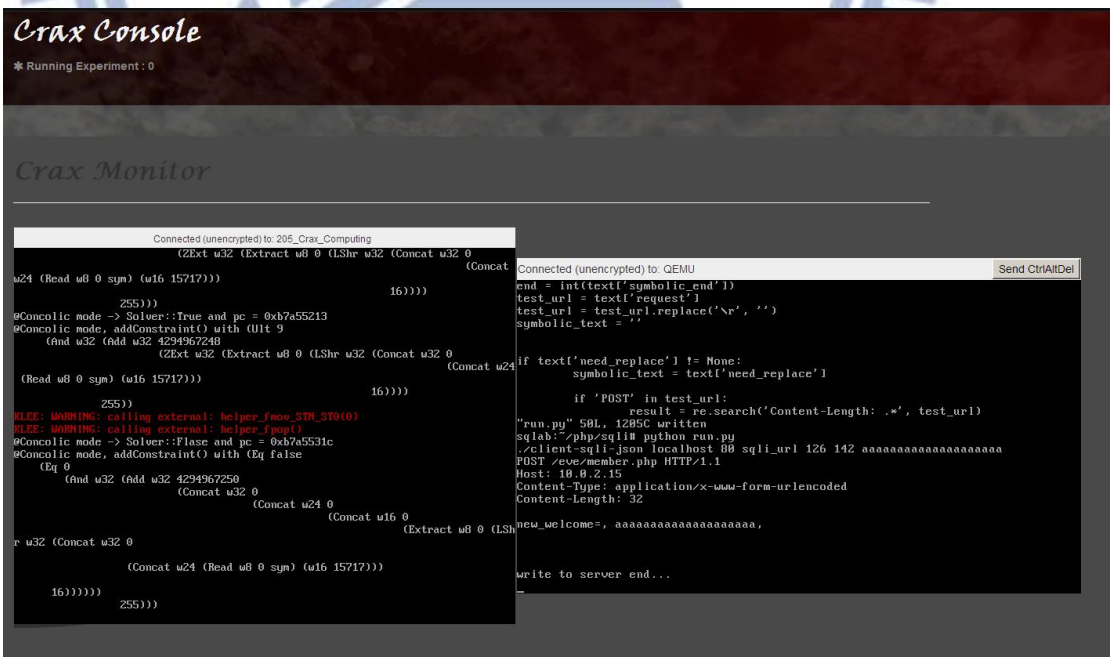
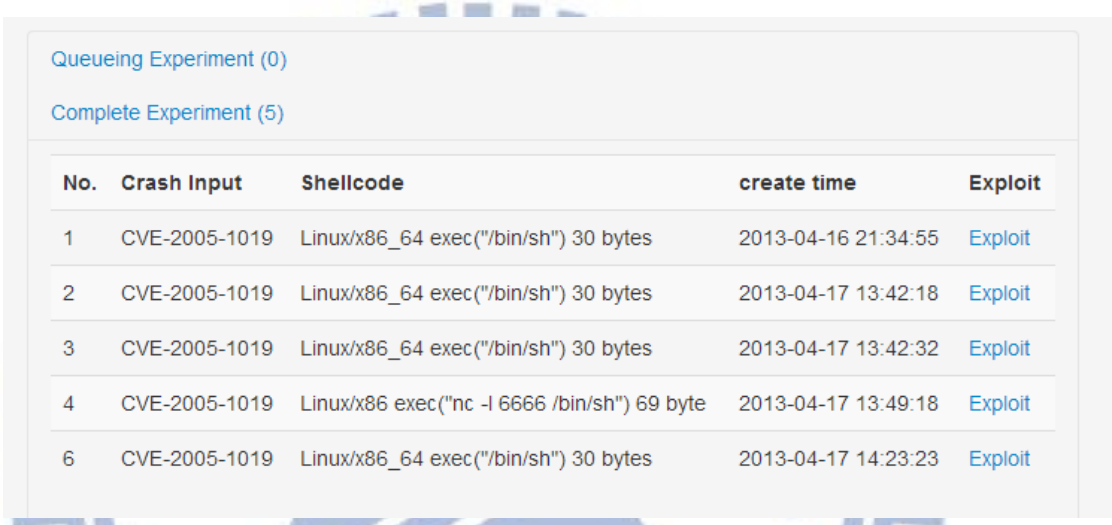


圖 4-4 運算節點直接控制介面

4-2-4 實驗記錄統計

控制端擁有 CRAX 實驗產生的紀錄檔，也詳細紀錄實驗每個階段的資訊。此實驗記錄頁面包含每個應用程式所產生過的脅迫、花費時間，以及實驗後的補充說明，以供日後追蹤修正。



Queueing Experiment (0)

Complete Experiment (5)

No.	Crash Input	Shellcode	create time	Exploit
1	CVE-2005-1019	Linux/x86_64 exec("/bin/sh") 30 bytes	2013-04-16 21:34:55	Exploit
2	CVE-2005-1019	Linux/x86_64 exec("/bin/sh") 30 bytes	2013-04-17 13:42:18	Exploit
3	CVE-2005-1019	Linux/x86_64 exec("/bin/sh") 30 bytes	2013-04-17 13:42:32	Exploit
4	CVE-2005-1019	Linux/x86 exec("nc -l 6666 /bin/sh") 69 byte	2013-04-17 13:49:18	Exploit
6	CVE-2005-1019	Linux/x86_64 exec("/bin/sh") 30 bytes	2013-04-17 14:23:23	Exploit

圖 4-5 CRAXUnix 脅迫產生紀錄

圖 4-6 CRAXWeb SQL Injection 偵測結果

4-3 雲端化 CRAX

由於 Libvirt 提供 API 管理虛擬機器，若要改用其他雲端服務如 EC2，只需修改 Libvirt API 成對應 EC2 API，即可移植至雲端主機上，執行 CRAX 實驗。

在原本的 CRAX 實驗，必須手動建立虛擬機器安裝能運行 S^2E 的主機端，同時能進行的實驗取決於安裝好的主機端數量。我們所設計動態增減運算節點的機制，可平行處理大量實驗，例如透過 CRAXUnix 檢查 EIP 是否被汙染，同時測試一千個檔案是否為惡意程式。

4-4 軟體弱點分析資料庫

不同於 CVE 只含問題描述、漏洞影響與危害程度，我們建置的 CRAX 實驗環境，是一個可執行的軟體弱點資料庫。新穎的攻擊手法或偵測、防禦方式均可直接從資料庫內找到對應執行環境，以進行測試。如表 4-4 列出與現行弱點資料庫的功能性差異。

透過 AutoCRAX 自動化產生脅迫，可以直接驗證軟體安全性問題，相較於手動產生脅迫，節省的時間達五六倍以上，對於操作人員的要求也較不嚴苛，只需要基本的 Unix 指令和標準化的實驗流程操作，不需要掌握各種複雜的攻擊手法(例如緩衝區溢位、格式化字串、指令碼植入等)。

	弱點資料	軟體	攻擊程式	執行環境	UI
CVE	完整	無	無	無	無
Exploit DB	參考 CVE	提供部分	提供	無	無
Metasploit	參考 CVE	無	提供	無	提供
AutoCRAX	參考 CVE	提供	提供	提供	提供

表 4-6 與現行弱點資料庫比較

4-5 0day 發掘、驗證及修補

透過 CRAX Console，快速建立弱點程式的執行環境，能測試最新版程式的安全性。以目前常見的圖書館自動化管理系統－Koha[19]為例，安裝過程複雜冗長，安裝頁面共十多面，所需安裝的相依性套件達數十個，但透過 CRAX-Init 控制頁面安裝一次後，即可重複使用。

我們檢視 Koha 上傳功能的程式，發現處理上傳壓縮檔時，檔案名稱的檢查不夠嚴謹，直接將使用者所提供的檔名傳入 Open 函式，因此只需要精心設計好檔案名稱，加入壓縮檔上傳，便能利用 Open 指令的 Pipe 功能，執行任意指令。

```
62      if ($suffix =~ m/zip/i) {      # If we were sent a zip file, process any
63          my ( $file, $filename );
64          undef $cardnumber;
65          $debug and warn "Passed a zip file.";
66          opendir my $dirhandle, $dir;
67          while ( my $filename = readdir $dirhandle ) {
68              $file = "$dir/$filename" if ($filename =~ m/datalink\.txt/i ||
69          }
70          unless (open (FILE, $file)) {
71          warn "Opening $dir/$file failed!";
72                  $errors{'OPNLINK'} = $file;
73          return $errors; # This error is fatal to the import of this directo
74          };
75
```

圖 4-7 Koha 問題程式碼片段

從 CRAX-Test 頁面直接開啟先前建置之環境，如圖 4-8。由於此攻擊為遠端攻擊，CRAX-Test 頁面只能觀察本機情況。必須使用另一台機器的瀏覽器，上傳有問題的壓縮檔，驗證我們發現的漏洞，如圖 4-9。

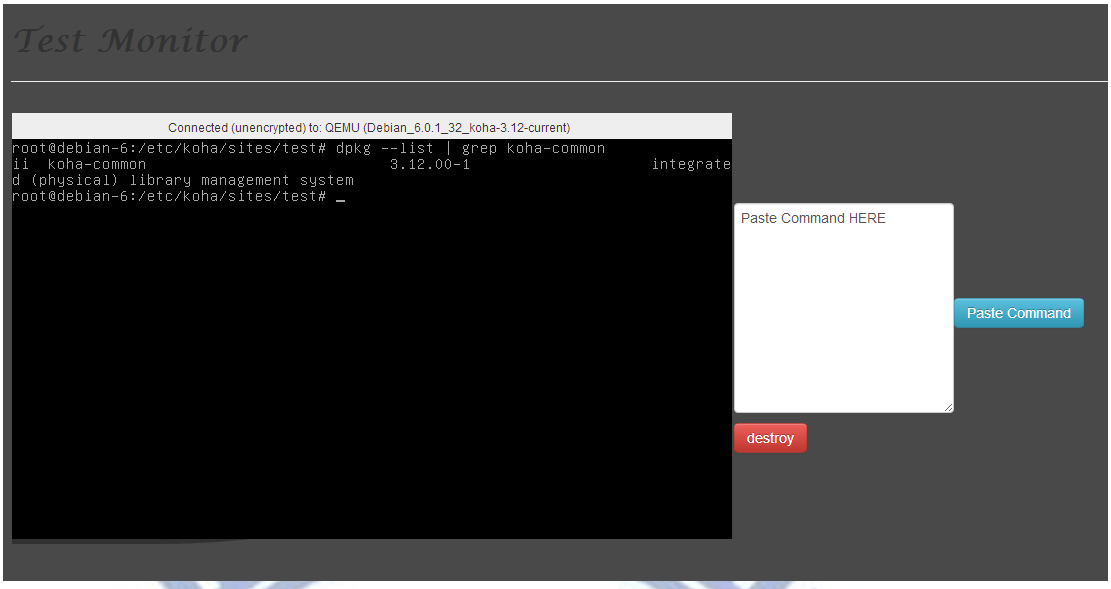


圖 4-8 CRAX-Test Koha-current 監控頁面

圖 4-9 Koha 圖片上傳指令植入漏洞

壓縮檔內植入 nc –e /bin/sh 140.113.208.227 12345 的指令，在 140.113.208.227 監聽 12345 埠，最後確認取得 Koha 機器的本機權限(圖 4-10)，驗證此漏洞，確實可執行任意指令。

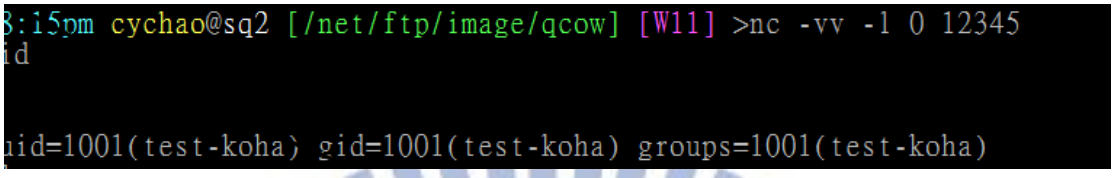


圖 4-10 Koha 本機權限獲取

驗證後，便可利用 CRAX-Test 的銷毀功能，刪除本次實驗的資料。刪除後若需再次測試，只要再次選取所需環境，即可再次驗證漏洞或嘗試漏洞修補。

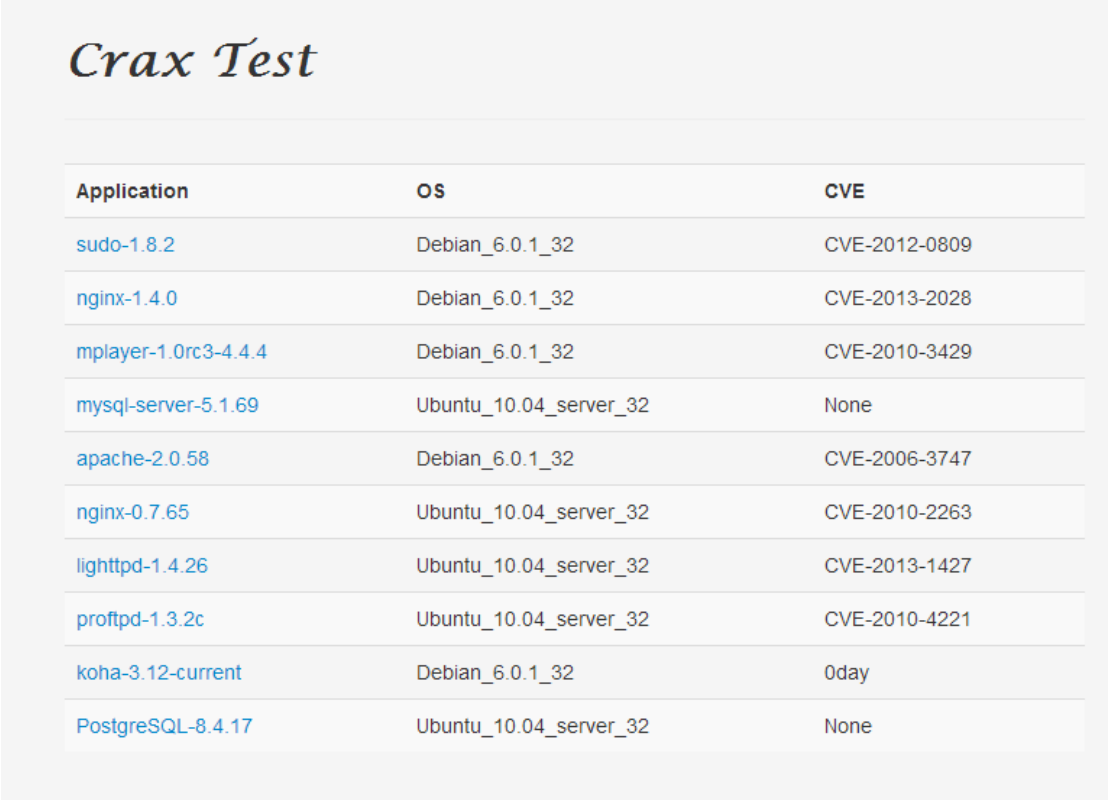


圖 4-11 CRAX-Test 執行環境選擇

第五章 結論與未來展望

5-1 結論

透過我們的系統，快速將有漏洞的程式儲存成可執行的環境，避免隨著時間過去，作業系統與編譯器更新，環境不同後，舊版程式無法正確編譯、安裝和執行的困境，所儲存的執行環境可以提供各種用途：漏洞驗證、修補、惡意程式偵測和 CRAX 中的 Exploit Generator 與 Fuzzer。

而現在軟體測試漸漸朝向雲端發展，我們這個基於雲端技術所建立的軟體弱點資料庫，透過雲服務可以二十四小時同時進行各種實驗，自動發掘軟體弱點，並將真實的軟體漏洞轉換為 wargame，作為教育訓練使用。

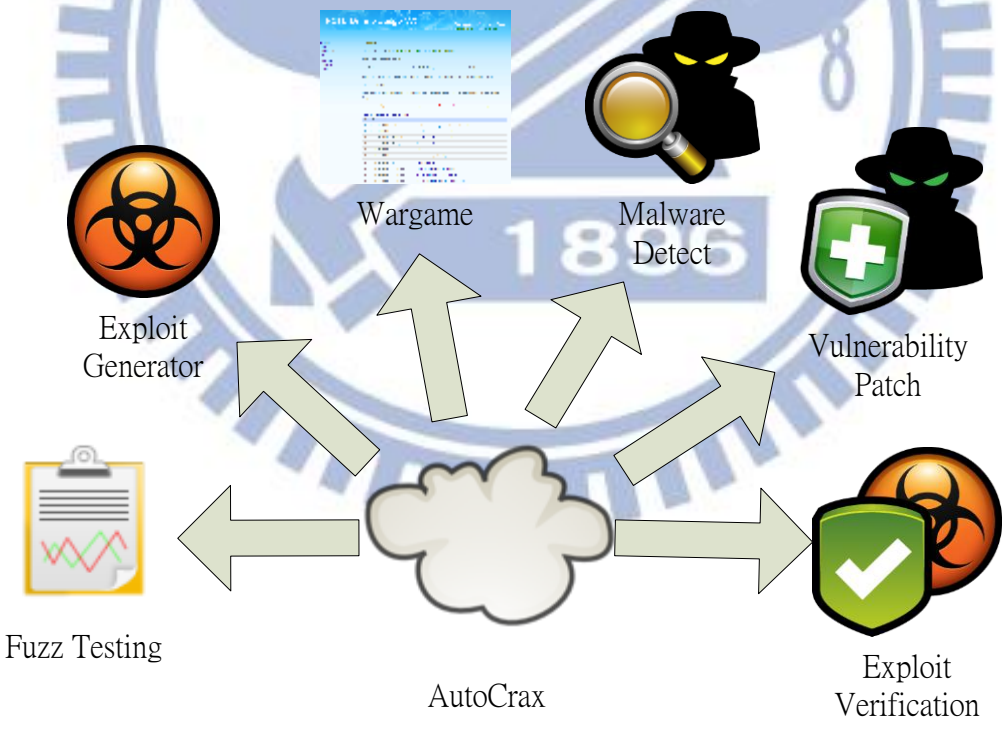


圖 5-1 AutoCRAX 功能

5-2 未來展望

為避免一個映像檔被多個 CRAX 同時開啟會造成不可預期的錯誤，目前的機制是啟動運算節點時一併複製映像檔，導致啟動一個新的運算節點時必須花費較多的時間，未來儲存映像檔的機制可以引入 ZFS 檔案系統差異備份的功能，來達到快速複製及節省空間的效果。ZFS 差異備份是在複製檔案時，共享儲存空間，當檔案被更動時，才儲存不同的部分，但因目前 ZFS 在大量檔案讀寫時效能會變得非常差，故還需要等待未來 ZFS 較穩定時才能導入。

抑或是加強 CRAX-Init 的功能，在安裝受測程式部分能在第一次安裝後記錄安裝步驟及備份所需檔案，此方式雖然會花費較多時間在初始化環境上，但可節省空間，只需儲存相關的設定檔和應用程式本身。將兩種方法合併使用可將較常測試的環境使用映像檔快照的方式，較少測試的環境則用加強後的 CRAX-Init 儲存，達到最好的效益。

參考文獻

[1] Farwell, J.P. and R. Rohozinski, *Stuxnet and the future of cyber war.* Survival, 2011. **53**(1): p. 23-40.

[2] King, J.C., *Symbolic execution and program testing.* Communications of the ACM, 1976. **19**(7): p. 385-394.

[3] Schwartz, E.J., T. Avgerinos, and D. Brumley. *All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask)*. in *Security and Privacy (SP), 2010 IEEE Symposium on*. 2010. IEEE.

[4] Sen, K. *Concolic testing*. in *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering*. 2007. ACM.

[5] Cadar, C., D. Dunbar, and D. Engler. *KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs*. in *Proceedings of the 8th USENIX conference on Operating systems design and implementation*. 2008.

[6] Lattner, C. and V. Adve. *LLVM: A compilation framework for lifelong program analysis & transformation*. in *Code Generation and Optimization, 2004. CGO 2004. International Symposium on*. 2004. IEEE.

[7] Chipounov, V., V. Kuznetsov, and G. Candea, S^{2}E*: A platform for in-vivo multi-path analysis of software systems.* ACM SIGARCH Computer Architecture News, 2011. **39**(1): p. 265-278.

[8] Bellard, F. *QEMU, a fast and portable dynamic translator*. 2005. USENIX.

[9] Shih-Kun, H., H. Min-Hsiang, H. Po-Yen, L. Chung-Wei, L. Han-Lin, and L. Wai-Meng. *CRAX: Software Crash Analysis for Automatic Exploit Generation by Modeling Attacks as Symbolic Continuations*. in *Software Security and Reliability (SERE), 2012 IEEE Sixth International Conference on*. 2012.

[10] CVE. *Common Vulnerabilities and Exposures*. Available from: http://cve.mitre.org/.

[11] Exploit-DB. *Exploits Database by Offensive Security*. Available from: http://www.exploit-db.com/.

[12] Metasploit. *Penetration Testing Software*. Available from: http://www.metasploit.com/.

[13] NCTU.Online.Judge. *使用者網路攻防能力評估系統(Wargame)*. Available from: http://wargame.cs.nctu.edu.tw/.

[14] Chen, L., *Construction of the New Generation Network Security Testbed—Testbed@ TWISC: Integration and Implementation on Software Aspect.* Institute of Computer & Communication, National Cheng Kung University, Tainan, Taiwan, 2008.

[15] 劉旭哲，*互動式線上教學之互動點推薦機制研究; A Recommending Mechanism for SettingInteractive Point in Interactive E-learning*. 2012.

[16] 梁偉明, *自動化網頁測試與攻擊產生*, in *網路工程研究所* 2012, 國立交通大學碩士論文: 新竹市. p. 35.

[17] 黃博彥, *自動產生攔截控制流程之攻擊程式碼*, in *資訊科學與工程研究所* 2011, 國立交通大學碩士論文. p. 54.

[18] 許基傑, *藉由選擇性符號操作執行之Android APPs隨性測試*, in *資訊科學與工程研究所* 2012, 國立交通大學碩士論文. p. 40.

[19] Koha. *Koha is the first free and open source software library automation package (ILS).* Available from: http://koha-community.org/.