殭屍網路特徵與分析

The Features of BotNet and its analysis

指導教授：張燕光; 專題成員：黃育笙、邢益城

開發工具：GCC、python; 測試環境：Linux ubuntu 6.0

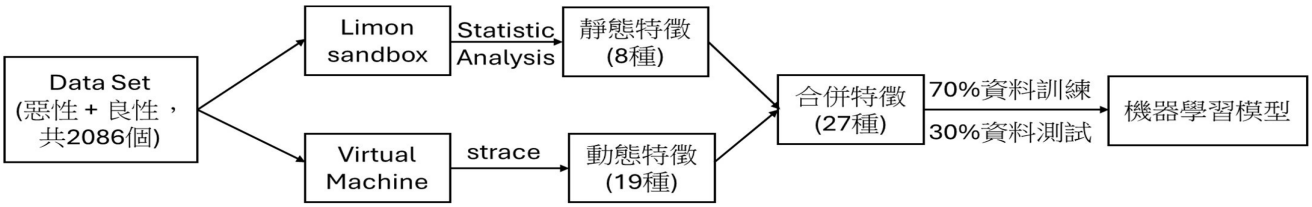
一、簡介：

在閱讀完學長的碩士畢業論文[1]後，我們希望在原作者的方法基礎上進行改進。當我們嘗試擷取機器學習所需的特徵時，可以將特徵分為靜態和動態兩類。靜態特徵與樣本的規格（如檔案大小）有關，不需要實際執行即可獲得，本次專題中我們使用 Limon Sandbox[2]的靜態分析功能來獲取靜態特徵。動態特徵的獲取則需要實際執行惡意軟體並計算其系統呼叫（system call）數量，我們利用 Linux 內建的 strace 指令來獲取動態特徵，請參考圖一。

我們遇到的主要問題是，惡意軟體不一定會表現出其真正的惡意行為。例如，殭屍網路病毒可能因為沒有接收到來自駭客的命令而一直處於等待狀態。為了解決這個問題，我們參考了著名殭屍網路 Mirai 的源代碼[3]，自製了一個殭屍網路，並以此作為惡意樣本。由於該樣本是我們撰寫的，因此我們可以透過各種命令來確保其表現出各種惡意行為。這樣就可以彌補機器學習訓練集不夠全面的問題。此外，原論文並沒有提到擷取 sendto 和 recvfrom 等系統呼叫作為特徵，因此我們新增了這兩個特徵，並測試其對準確率的影響。

我們實作的殭屍網路（botnet）如圖二「殭屍網路運作架構圖」所示。當它開始執行時，會等待我們自行架設的指揮與控制（C&C）伺服器的命令。此時，它會開始掃描 /proc/a_PID 資料夾底下所有進程（process）的信息，如命令行參數、環境變量、資源使用情況等，以達到監視系統的目的[4]。

在收到來自 C&C 伺服器的命令後，會利用 fork 創建一個子進程（child process）並發起 DDoS 攻擊，發送大量的垃圾 UDP 封包。在這階段會大量使用 sendto 這個系統呼叫（system call），這也是為什麼我們需要新增 sendto 作為特徵以改進模型準確率。



圖一: 擷取動、靜態特徵的流程圖

資料集介紹:

名稱	種類	樣本數量	來源
bin	良性	428	Linux 系統內建
sbin	良性	197	Linux 系統內建
Linux Malware	惡性	464	[5]
MalwareBazaar Database	惡性	997	[6]

靜態特徵列表: 檔案規格

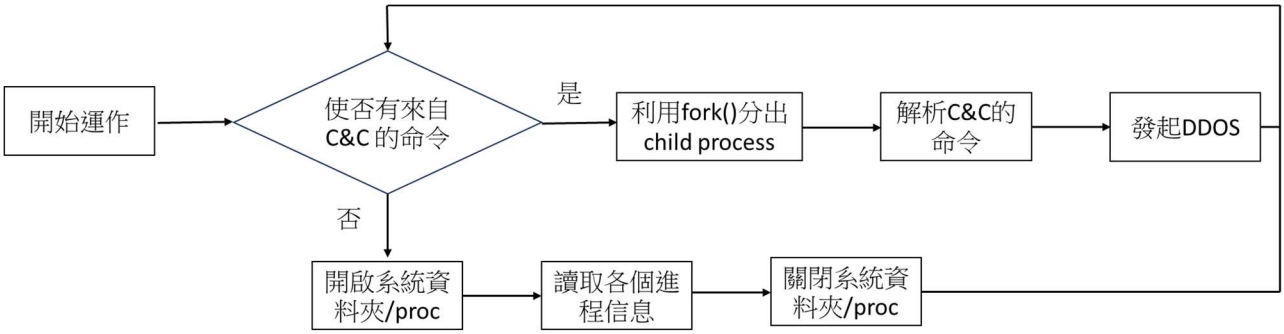
以下是原論文[1]作為靜態特徵的所需的檔案規格:

Number of Program Headers	Section header string table index	Size of .data Section
Number of section headers	Size of .text Section	Size of .rodata Section
File Size	Size of .bss Section	

動態特徵列表: 系統呼叫(system call)

以下是原論文[1]作為動態特徵的 system call:

execve	dup	creat	unlink	rename	write	bind
read	socket	fork	dup2	pipe	connect	
close	kill	open	clone	accept	chdir	



圖二: 殭屍網路運作架構圖

二、測試結果：

模型名稱	Accuracy	Precision	Recall
Decision tree(改進前)	95%	97.5%	96.3%
Decision tree(改進後)	95.8%	98%	96.76%
Logistic regression(改進前)	79%	78.2%	99.35%
Logistic regression(改進後)	80.9%	80.2%	98.27%
KNN(改進前)	94%	98.1%	98.35%
KNN(改進後)	94.3%	98.5%	94%
Random Forest(改進前)	95.8%	98%	96.3%
Random Forest(改進後)	96.2%	97.6%	96.9%
SVM(改進前)	74.2%	74.4%	99.35%
SVM(改進後)	74.3%	74.3%	99.1%
Neural network(改進前)	92.6%	98.1%	91.5%
Neural network(改進後)	93%	98.6%	92.2%

可以發現在我們改進特徵擷取的方式後，各模型的準確率確實有所提升。由於此次任務是辨認惡意軟體，因此低偽陰性（即高召回率）變得尤為重要，因為偽陰代表實際上為惡意但卻被模型辨認為良性的次數。在考慮到高召回率且同時具有較高的準確率和精確度的情況下，我們可以發現決策樹（Decision Tree）、K 近鄰（KNN）、隨機森林（Random Forest）和神經網絡（Neural Network）最適合本次任務。

三、參考文獻：

[1] 黃志翔 “Detecting malware distributed by IoT Botnet based on Machine Learning and Deep Learning”,成大資訊碩士論文, 2023.01

[2]” Limon - Sandbox” 2015, https://github.com/monnappa22/Limon

[3]” Mirai-Source-Code” 2015, https://github.com/jgamblin/Mirai-Source-Code/tree/master/mirai/bot

[4]” The /proc Filesystem” 2009, https://docs.kernel.org/filesystems/proc.html

[5]” Linux Malware” 2014, https://github.com/MalwareSamples/Linux-Malware-Samples

[6]” MalwareBazaar Database” 2023, https://bazaar.abuse.ch/browse/