運用深度學習技術於網站 Webshell 檢測

柯宏叡

國立中興大學資訊科學與工程學系

曾學文

國立中興大學資訊科學與工程學系

王旭正*

中央警察大學資訊管理學系

* whom correspondence: dopwang@gmail.com

摘要

網際網路的快速發展大大改變了人們生活和工作方式，隨著各種網站服務的推陳出新，網路安全也逐漸獲得愈來愈多的關注。許多網站被植入 websell 而不自知，不僅導致資料外洩，更可能成為網路攻擊的幫兇。然而 Webshell 的變化極多，偵測相當不易，故本文就最常見的以 PHP 為主的 Webshell 來研究，運用字頻分析與深度學習來進行檢測與自動識別。本文提出的方法可避免於以往複雜的人工處理方式，僅透過訓練就能達到高準確率。我們的實驗結果也顯示，這種方法能不用透過轉換，就能達到高準確率，誤判率也較其他研究更低。

關鍵詞: Webshell、PHP、深度學習、字頻分析

壹、前言

愈來愈多的現代人已與網路密不可分，生活上的食衣住行的所需服務都是透過網路來完成。然而多元化的網站服務讓人們享受到便利的生活，也因此許多重要的資料不可避免地都儲存在網站之中，引起許多惡意攻擊者的注意，導致近年來對於網站的攻擊與入侵有增無減。由於近來的攻擊更趨於隱密，攻擊者在完成入侵後，不一定會去修改網站首頁，反而是留下後門讓之後隨時都可以入侵網站竊取重要資料。其中最常見的後門就是植入 Webshell 程式，一旦 Webshell 被順利上傳至網站，並取得高權限執行，網站就相當於落入攻擊者的手上。

webshell 主要是由網頁程式語言編寫而成，例如：ASP、PHP、JSP 等，通常包含一系列攻擊功能，例如文件上傳和下載、指令執行、查看系統狀況與權限提升等。為了避免被系統管理者發現，Webshell 通常混入到許多一般網頁或嵌入到其他網頁中，因此不易查覺。

對 Webshell 的檢測是一項困難的任務[1]，原因在於：

1. 惡意的 Webshell 使用的功能與網站管理員用來遠端管理自己的伺服器的功能相似。
2. Webshell 一般不會在系統日誌中留下完整的記錄，攻擊者並不會持續使用 Webshell，這使得網站管理者很難藉由流量分析來發現異常行為。
3. Webshell 的長度大小不同，從一行指令到具有複雜的圖形化使用者介面都有。
4. Webshell 可以使用不同語法並混合不同語言（如 HTML）編寫。
5. Webshell 可運用字串編碼來繞過規則與特徵比對工具。

而對於 webshell 檢測方法，目前有以下幾種類型的方法：

1. 靜態檢測：不直接操作 Webshell，而是僅就目前的網頁內容來尋找是否有 Webshell 常使用的字詞，如 eval、assert、system 等，再就這些網頁進一步分析為具惡意功能的 Webshell[2]。例如在 Linux 環境中使用 grep 指令，或在 Windows 環境中使用 findstr 指令來找尋特定的文字。這種方法檢測速度快，但需要對 Webshell 有相當的背景知識者才有辦法一步步的找出惡意程式的存在。此外，所能尋找字詞特徵較為固定，容易被有心人透過變化字串方式的方式來避免被檢測出來。
2. 動態檢測：在沙箱（sandbox）或是安全的環境中執行 Webshell，檢測系統的更動情形來確定是否為惡意程式。透過 HTTP（Hyper Text Transfer Protocol）連線操作 Webshell 會留下具有特徵的連線紀錄，網站管理者就能運用入侵偵測系統來偵測或阻擋 Webshell[3]。但為了建立沙箱或安全的環境需要更多的資源，且 Webshell 可被嵌入到正常的網頁中，需透過特定條件，例如輸入特定關鍵字才會啟動，因此不易分析連線特徵。
3. 統計方法：透過統計文件中出現字詞、函數、特殊符號等所佔比例來評估是否有文件異常，這種檢測的方法較靜態檢測更有效，但對於被刻意混淆（Obfuscation）的程式碼的還是容易誤判。
4. 深度學習（Deep Learning）：將字詞透過詞向量（Word embedding）技術轉為以數值或矩陣方式表示，再運用深度學習的相關方法進行訓練與測

試，最終建立高準確度的模型來判斷檔案是否為 Webshell。深度學習透過自動建立特徵的方式來學習判斷，這種方法可達高準確率且無需先擷取特徵，使攻擊者的惡意 Webshell 較難以透過混淆程式碼來藏匿。缺點是需事先建立相關的資料集，且訓練判斷的模型時通常需要大量的運算資源。

以上這些方法都各有優缺點，如分析網站連線紀錄或是檢測網頁程式碼，不僅需要專業的知識，一旦網站規模龐大，要檢測大量的連線紀錄或網頁程式碼也需要相當多時間投入方能達成。統計方法雖然有效，但也需要找出特徵後，才能依據這些特徵來進行統計分析。因此我們希望透過深度學習的技術來自動化建立特徵，透過訓練提高模型的判斷精確程式，就能更快且更有效的找出 Webshell。相關方法的比較如表 1 所示。

表 1：各項 Webshell 檢測方法比較

方法	優點	缺點
靜態檢測	檢測速度快	需要足夠的背景知識進行判斷
動態檢測	判斷 Webshell 的準確度高	需額外的資源建立檢測環境
統計方法	較靜態檢測方式佳	容易誤判
深度學習	無需先擷取特徵，準確度高	需先有足夠的資料集進行訓練

依據 w3techs 公司就全球網站所用程式語言的統計資料而知[4]，PHP 佔了所有網站的 79.1%，遠高於 ASP.NET（8.4%）、Ruby（4.9%）及 Java（3.6%）。所以我們也在樣本蒐集中發現許多 Webshell 是以 PHP 為主，另由於 ASP 也因多為老舊的程式碼，現今也不易取得足夠的樣本，所以本文的研究標的以 PHP 為主。本文結構編排如下:在第二節將會介紹相關研究的背景知識；第三節就相關文獻進行探討；在第四節提出我們的實驗設計及評估杞準；於第五節針對實驗結果進行討論，同時將所得到結果進行整理與說明，最後於第六節提出研究結論。

貳、背景知識

不論是使用 PHP、ASP 或 JSP 撰寫的網頁或 Webshell，都並非經過編譯（Compile）產生，其本質上就相當於是文件，且具有一定之規則，如一句話木馬的程式碼如下：

<?php eval($_GET[cmd]); ?>

就在網站的 PHP 網頁中僅有短短的一串文字，就能讓攻擊者以參數 cmd 來對網站下達各種指令。因此我們可以將程式語言也視為是一種自然語言（Natural Language），透過語言的分析來判斷文件是否是 Webshell 還是正常的網頁。

一、自然語言處理

在自然語言與 Webshell 的相關研究中，最常見的應用是詞袋（Bag-of-words）、TF-IDF(Term Frequency–Inverse Document Frequency)及 word2vec。

1. 詞袋：在詞袋模型下，一段文字（比如一個句子或是一整篇文章）可以用一個矩陣來達示。這種表示方式不考慮文法以及詞的順序，目前被廣泛應用在文件分類上，但部分研究認為詞袋模型可能無法有效分辨被混淆過的 Webshell。
2. TF-IDF：其中 TF 意思為詞頻，就是某個字詞出現於文章中的頻率。IDF 表示反向文章頻率，用來衡量某個字詞於所有文章中的重要性。最後將 TF 乘以 IDF，就能獲得每個字詞的權重。
3. Word2vec: Word2vec 為 Google 於 2013 年所提出的自然語言處理工具[5]。Word2vec 主要分為 CBOW（Continuous Bag of Words）和 Skip-Gram 兩種模式。CBOW 是從原始語句推測目標字詞；而 Skip-Gram 正好相反，是從目標字詞推測出原始語句。運作方式為針對文件中的字詞來進行向量轉換，經過訓練完成之後，word2vec 模型可用來將每個詞映射到一個向量，就可用來表示詞對詞之間的關係。若兩個字詞距離愈近，表示這兩個字詞在文章中一起出現的可能性就愈高。

二、深度學習

深度學習可以將特徵擷取自動化，快速進行辨別，能較以往採用統計分析的方法更有效率。在應用於 Webshell 分析，最常被使用的就是卷積神經網絡(Convolutional Neural Network, CNN)與 LSTM（Long Short-Term Memory）這兩種深度學習模型，分述如下：

一、CNN 主要可分為：

1. 卷積層（Convolutional layers）：卷積類神經網路的設計概念，就是針對資料特性判斷鄰近的資料；若為二維資料型態，鄰近的資料就包括上下左右的平面方向。若是一維資料，就是要建立起資料前後的特徵。

2. 池化層（Max-pooling layer）: 經過卷積層運算後，數值高者表示為重要特徵。雖然得到位置資訊後便可以直接進行下一步判斷，但實際上不一定需要那麼多的訊息，因此衍生出名為池化層的結構。池化層將數值分區化簡，亦即與特徵不符合的區域合併省略，藉此減少大量不必要的運算，並可搭配卷積層重覆執行，精簡產生所輸入的資料特徵。
3. 全連接層（Fully Connected layer）：全連接層相當於分類器，把我們經過數個卷積、池化後的結果進行分類。而通常分類所使用的方法為 softmax，並並以 crossentropy 做為損失函數。

二、LSTM：

Hochreiter 和 Schmidhuber 於 1997 年提出的長短期記憶模型 (Long Short-Term Memory，LSTM)[6]，在神經單元中加入了遺忘門、輸入門以及輸出門三個步驟，改進原本的 RNN（Recurrent Neural Networks）模型無法保留長期記憶的缺點，提高了其在長期記憶的表現。LSTM 主要可分為：

1. 輸入門（Input Gate）：控制目前輸入訊息是否融入細胞狀態。當前輸入的訊息可能是重要的，也可能不重要，輸入門的目的就是判斷當前輸入的訊息對全局的重要性。這個門是 Sigmoid 函數，因為它是二元分類的函數，表示要加入與否。當開關打開的時候，網路將排除目前輸入的訊息。
2. 輸出門（Output Gate）：輸出門的目的是從細胞狀態產生隱藏層單元。並不是所有的訊息都和隱藏層單元有關，因此，輸出門的作用就是判斷哪些部分是有用的，哪些部分是無用的，這個門也是使用 Sigmoid 函數。
3. 遺忘門（Forget Gate）：控制上一時刻細胞狀態的訊息融入細胞狀態。當前輸入的訊息可能為延續之前的訊息，也可能是新的內容。和輸入門相反，不對當前訊息的重要性作判斷，而判斷的是上一時刻的細胞狀態對計算當前細胞狀態的重要性。這個門通常是 Sigmoid 函數，表示過往的資訊是否被遺忘或記憶。當開關打開的時候，網路將不考慮上一時刻的細胞狀態。
4. 細胞狀態（Cell State）：綜合了當前訊息和前一時刻細胞狀態的訊息，透過使用 tanh 函數來判斷是否移除長期記憶。

兩種深度學習模型在訓練後判斷 Webshell 上都有相關的研究，惟 LSTM 的訓練時間遠高於 CNN，若未能獲得更好的結果時，研究上就會傾向於使用 CNN 為主。

參、相關文獻探討

在 2019 年時，Tao et al.[7]提出以先就 Webshell 進行去除雜訊，提取攻擊部分的資料（Malicious Payload），再以 TF-IDF 的方式將這此資料轉為詞向量。接下來，並分別採用多層感知機（Multi-Layer Perception, MLP）、CNN、LSTM 等深度學習技術來檢測，結果顯示 CNN(Convolutional neural network)與 LSTM(Long Short-Term Memory)的準確率（Accuracy）分別達到 97.93%與 97.12%，而 MLP 則能達到更高的準確率 99.57%，但所費時間遠高於 CNN 與 LSTM，且可能會有 over-fitting 的問題出現。然而 Tao et al.的論文中，對於相關所使用的樣本，僅提到來自 Github，但卻未明列出樣本數與相關的網址等訊息。

為了要處理 PHP 檔案可能會受到混淆程式碼的處理，因此有些研究就導入使用 Zend 的 VLD（Vulcan Logic Disassembler）套件來處理 PHP 程式。VLD 的這一項套件能將 PHP 程式轉為操作碼（Opcode），也就是 PHP 的執行單元（Execution Unit），藉此可減少混淆後的影響，並簡化要分析字詞。如 Kang et al.[8]將 PHP 先轉為操作碼，再以 TF-IDF 進行轉換後，再以所調整後的模型 RF-AdaCost 進行訓練與分類判斷，使準確率達到 95.3%。

在 Word2vec 的應用上，Lv et al. [9]於 2019 年採用 Word2vec 進行 Webshell 分析，發現準確率僅約 70%，其原因在於直接使用 word2vec 於 Webshell 上，其轉換時會將其他雜訊，如說明或註解資料也一併轉換，導致準確率偏低。因此 Zhang et al.[10]於 2020 年改以先將 PHP 程式轉為操作碼，並分別運用 TF-IDF 及 Word2vec 轉換為向量，再以 XGBoost(eXtreme Gradient Boosting package)來進行分析檢測，發現 Word2vec 的準確率能達 98.54%，還高於使用 TF-IDF 的 96.49%。NGUYEN et al.[11]則是使用了 Yara 這套惡意程式分析工具來先判斷 PHP 網頁程式碼是否為惡意程式，再將 PHP 轉換為操作碼，去除重複資料後以 CNN 深度學習模型進行訓練，使準確率達到 99.02%，惟未提及是用何種方式將字詞轉為向量。在 2021 年時，Zhou et al.[12]使用操作碼與 Word2vec 來進行，採用 LSTM 技術，準確率可達到 95.2%，但其他的相關評估標準，如召回率等則付之闕如。近來，則有 Al et al.[13]等提出以集成學習（Ensemble Learning）為基礎的偵測方式 WL-LSMR。WL-LSMR 的方法是採用邏輯回歸（Logistic Regression）、支援向量機（Support Vector Machine）、MLP 與隨機森林（Random Forest）等演算法進行組合，並配合使用操作碼來進行分類訓練與判斷，其實驗顯示準確率可達 94.19%，召回率亦可達 99.14%。但使用集成學習的缺點也在於因訓練時使用了多種演算法，訓練所需時間較久，且 WL-LSMR 的實驗中，正常樣本為 5379 個，但 Webshell 樣本僅使用了 566 個，Webshell 樣本數相對偏少。

我們由相關文獻可知因操作碼能減少 PHP 程式中的雜訊，所以被不少研究納入使用，但是操作碼僅適用於 PHP 程式語言，並需搭配 VLD 套件方能順利運作，並非所有程式語言皆可進行轉換，且為了要能在不同平臺上進行檢測，這些平臺也要同步安裝轉換工具才能運行，這也使得轉換操作碼的方法無法在未來應用到以其他程式語言撰寫的 Webshell。因此，本文提出基於詞袋與 CNN 的檢測方法，要能不預先轉為操作碼的情況下，也能達到高準確率。

肆、實驗設計及評估

一、樣本

所用來檢測的資料規劃如下：

- Webshell：github 上使用者蒐集的 Webshell 資料，來源如表 2 所示，PHP 檔案數共計 4739 個。
- 正常樣本：以開放原始碼的系統為主，來源如表 3 所示，PHP 檔案數共計 6925 個。

表 2：github 上的 Webshell 來源

編號	網址
1	https://github.com/tennc/webshell
2	https://github.com/bartblaze/PHP-backdoors
3	https://github.com/b374k/b374k
4	https://github.com/JohnTroony/php-webshells
5	https://github.com/xl7dev/webshell
6	https://github.com/BlackArch/webshells
7	https://github.com/fuzzdb-project/fuzzdb
8	https://github.com/LuciferoO/webshell-collector
9	https://github.com/ysrc/webshell-sample
10	https://github.com/webshellpub/awsome-Webshell
11	https://github.com/lcatro/PHP-Webshell-Bypass-WAF
12	https://github.com/linuxsec/indoxploit-shell

表 3：open source 的程式檔案來源

編號	名稱	網址
1	Word Press	https://github.com/WordPress/WordPress
2	Joomla	https://github.com/joomla/joomla-cms
3	Laravel	https://github.com/laravel/laravel
4	phpMyAdmin	https://github.com/phpmyadmin/phpmyadmin
5	phppgadmin	https://github.com/phppgadmin/phppgadmin
6	phpBB	https://github.com/phpbb/phpbb

二、流程：

1. 樣本取得：由上述的網址，從 github 上取得相關 Webshell 及正常的 PHP 檔案。
2. 去除雜訊：為避免影響詞袋建立，PHP 檔需將註解資料與 HTML 標記移除，來獲取 PHP 程式碼的部分，如圖 1 所示。
3. 向量化處理：將去除雜訊後的 PHP 檔以詞袋方式轉換為向量。
4. 樣本分割：將樣本分割為訓練與測試資料集，比例為 6 比 4。
5. CNN 運算：使用 CNN 來進行訓練與驗證，並設定 epoch 為 500，提前結束的條件為驗證準確率達 0.998。
6. 結果顯示：計算出準確率等相關結果。

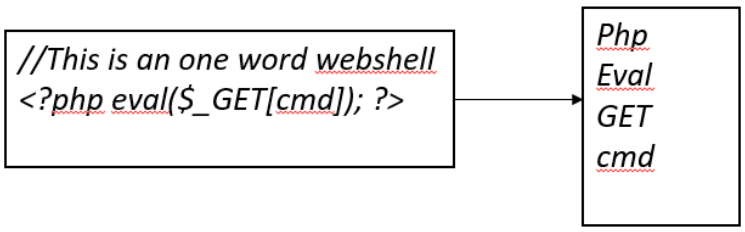

圖 1：去除雜訊

三、CNN 網路

我們採用 CNN 做為分類演算法，架構如圖 2 所示，主要可分為：

1. 卷積層：我們採用了三個卷積層進行特徵收集，filter size 分別為 14,15,16， 輸出維度設為 128，activation function 使用 Relu。
2. 池化層: 卷積層的資料送到池化層，以最大值來保留重要特徵。
3. 全連接層：以 softmax 進行分類，並以 crossentropy 做為損失函數。

4. 設定 dropout 函數為 0.5 來避免 overfitting，設定優化器為 Adam，學習率為 0.001。

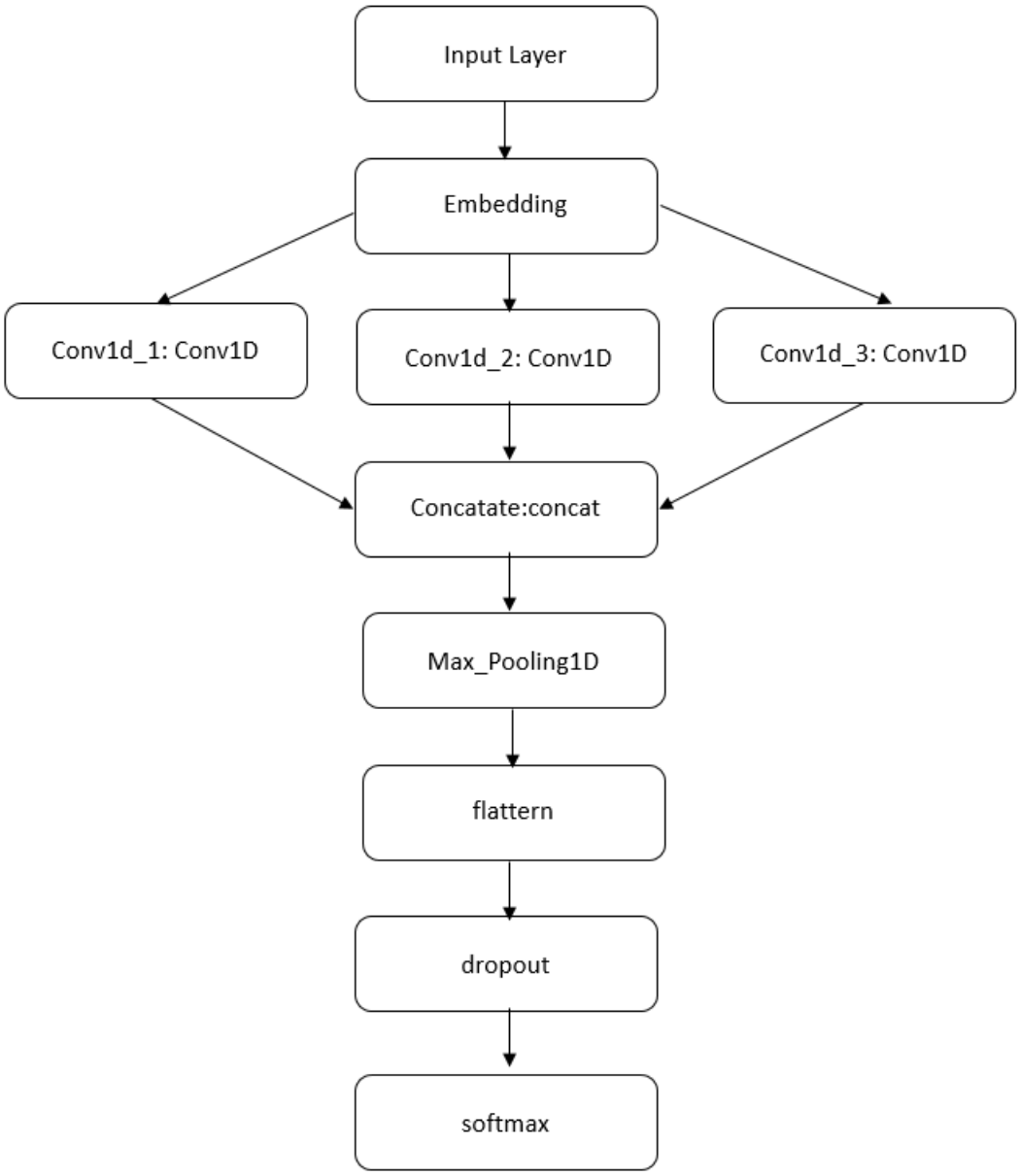

圖 2：本實驗的 CNN 架構

四、評估標準

Webshell 的分類檢測，我們透過觀察：

- TP (True Positive) —實際為 True，預測為 Positive。預測的結果與實際情況相同，稱為真陽性。
- TN (True Negative) —實際為 False，預測為 Negative。預測的結果與

實際情況相同，稱為真陰性。

- FP (False Positive) —實際為 False，預測為 Positive。預測的結果與實際情況不同，稱為假陽性。
- FN (False Negative) —即實際為 True，預測為 Negative。預測的結果與實際情況不同，稱為假陰性。

並據以計算準確率(Accuracy)、精確率(Precision)、召回率(Recall)、平衡 F 分數(F1_score)與誤判率(False Positive Rate, FPR)，計算公式如下：

準確率的定義為正確分類的樣本（*TP* 及 *TN*）與總樣本的比率。

$$Accurary = \frac{TP+TN}{TP+FP+FN+TN} \quad (1)$$

精確率的定義為真陽性除以真陽性與假陽性相加的比率。

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

召回率被定義為真陽性除以真陽性與假陰性之和的比率。

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

F1_score 用來衡量分類模型精確度的一種指標。它同時兼顧了分類模型的精確率和召回率，可視為是精確率和召回率的調和平均。

$$F1\_score = \frac{2TP}{2TP+FP+FN} \quad (4)$$

FPR 為假陽性除以假陽性與真陰性之和。

$$FPR = \frac{FP}{FP+TN} \quad (5)$$

伍、實驗結果與討論

我們使用 PHP 正常樣本（White）共 6925 個檔案，PHP Webshell 3749 個檔案進行實驗，訓練與測試集的比例為 6:4。實驗的主機採用國家高速網路與計算

中心的 CDX 平台（https://cdx.nchc.org.tw ）上的 Ubuntu 18.04、CPU 使用 16 核、記憶體使用 32GB，實驗在 Epoch 43 時提前結束，訓練時間約 6 小時。

經計算出的相關評估標準與相關文獻比較如表 4 所示。可以見到我們的方法較其他的方法準確率更高，且在假陽性率上更低。相較於 Tao et al.同樣採用 CNN 的方法，我們將原本各個卷積運算後進行池化的方法，改為先疊合再進行池化運算的方式，來獲得更多的特徵，因而有更高的準確率。

表 4: PHP Webshell 實驗結果

評估指標	準確率 Accuracy	精確率 Precision	召回率 Recall	平衡 F1 分數 F1 score	假陽性率 FPR
Tao et al.[7]	97.93%	96.13%	99.34%	97.7%	N/A
Kang et al.[8]	95.3%	95.36%	95.06%	N/A	4.48%
Lv et al.[9]	99.5%	99.2%	99.7%	99.4%	N/A
Zhang et al.[10]	98.8%	99.75%	98.71%	N/A	N/A
Nguyen et al.[11]	99.02%	99.04%	99.02%	99.03%	1.47%
Zhou et al.[12]	95.2%	N/A	N/A	N/A	N/A
AI al.[13]	94.28%	99.14%	N/A	N/A	N/A
我們的方法	99.83%	99.79%	99.72%	99.75%	0.21%

目前已有開放原始碼的惡意程式檢測工具，如 Shell Detector[14]與 PHP Malware Finder[15]等，但其中的 Web Shell Detector 已超過 7 年未更新，因此不納入比較。另外還有迪元素科技有限公司所開發的 D 盾[16]、百度公司所開發的 WEBDIR+[17]皆是有名的 Webshell 檢測工具。我們由 Webshell 樣本抽出 2787 個檔案與這些工具程式比較，結果如表 5 所示。我們可以見到 D 盾的表現較 php-malware-finder 與 WEBDIR+更好，但並沒能比我們的方法更好。我們再以正常的 PHP 樣本中抽出 6652 個進行檢測，依據相關評估標準如表 6 所示，我們也可見到我們的方法在多數的評估結果中表現最佳。

表 5: 以 Webshell 進行與其他檢測工具比較結果

工具評估	檢出檔案數	準確率	備註
D 盾	2583	92.68%	2021 年 7 月 5 日版本
php-malware-finder	2104	75.6%	

WEBDIR+	2447	87.8%	2021 年 8 月 2 日檢測
我們的方法	2774	99.53%	

表 6: 與其他檢測工具的評估標準比較

工具評估	準確率 Accuracy	精確率 Precision	召回率 Recall	平衡 F1 分數 F1_score	假陽性率 FPR
D 盾	97.69%	99.46%	92.68%	95.95%	0.21%
php-malware-finder	92.29%	97.95%	75.49%	85.26%	0.66%
WEBDIR+	96.39%	100%	87.8%	93.5%	0%
我們的方法	99.92%	99.89%	99.85%	99.87%	0.05%

陸、結論

面對越來越多的 Webshell 被植入到網站中，導致網站落入有心人手裡變成跳板或攻擊平臺，許多程式或方法也被開發出來，但面對惡意程式不斷變形，對於資安人員形成一大挑戰。我們在本文中提出了一個有效的方法，無須先將 PHP 轉為操作碼，而是將 PHP 原始程式碼去除雜訊後轉換為詞袋模型，運用深度學習 CNN 的技術，透過訓練來預測 PHP 檔是否嵌入了惡意程式碼。實驗結果證明我們所提出的方法達到了 99.83%的準確率，假陽性率為 0.21%。深度學習模型的訓練不需要複雜的人工進行特徵擷取，因此攻擊者難以透過既定的特徵來進行迴避，也就是說，CNN 等深度學習技術將可防止許多的未知 Webshell 攻擊。我們未來的目標是將我們的方法擴展到其他程式設計語言，如 ASP.NET、Java、Python 等，就能協助資安人員防止更多未知的 Webshell 攻擊。

參考文獻

[1] A. Hannousse and S. Yahiouche, “Handling Webshell attacks: A systematic mapping and survey,” Computers & Security, Volume 108, 2021.

[2] M. Ahmed, A. N. Mahmood, and J. Hu, “A survey of network anomaly detection techniques,” Journal of Network and Computer Applications, vol. 60, pp. 19 – 31, 2016.

[3] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, “Host-based intrusion detection system with system calls: Review and future trends,” ACM Computing Surveys (CSUR), vol. 51, no. 5, pp. 1–36, 2018.

[4] W3techs, “Usage statistics of server-side programming languages for websites”, https://w3techs.com/technologies/overview/programming_language, access: 2021-07-23.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” Proceedings of 2013 Advances in Neural Information Processing Systems(NIPS 2013), 2013.

[6] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997.

[7] F. Tao, C. Cao and Z. Liu, “Webshell Detection Model Based on Deep Learning,” Artificial Intelligence and Security. ICAIS 2019. Lecture Notes in Computer Science, Vol. 11635, pp.408-420, Springer.

[8] W. Kang, S. Zhong, K. Chen, J. Lai and G. Xu, “RF-AdaCost: Webshell detection method that combines statistical features and opcode,” Proceedings of the 3rd International Conference on Frontiers in Cyber Security(FCS 2020), Springer Singapore, Singapore, pp. 667-682, 2020.

[9] Z. H. Lv, H. B. Yan and R. Mei, “Automatic and Accurate Detection of Webshell Based on Convolutional Neural Network,” 15th International Annual Conference, CNCERT 2018, pp. 73-85, Beijing, China, August 14–16, 2018.

[10] H. Zhang, M. Liu, Z. Yue, Z. Xue, Y. Shi and X. He, "A PHP and JSP Web Shell Detection System with Text Processing Based on Machine Learning," 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1584-1591, 2020.

[11] N. H. Nguyen, V. H. Le, V.O. Phung and P. H. Du, “Toward a deep learning

approach for detecting PHP Webshell," Proceedings of the Tenth International Symposium on Information and Communication Technology, SoICT 2019, pp. 514-521, 2019.

[12] Z. Zhou, L. Li and X. Zhao, "Webshell Detection Technology Based on Deep Learning," 2021 7th IEEE Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS) , pp. 52-56, 2021.

[13] Z. Ai, N. Luktarhan, Y. Zhao and C. Tang, "WS-LSMR: Malicious Webshell Detection Algorithm Based on Ensemble Learning," in IEEE Access, vol. 8, pp. 75785-75797, 2020.

[14] Shell Detector, https://github.com/emposha/Shell-Detector, accessed on 2021/7/23.

[15] PHP Malware Finder, https://github.com/jvoisin/php-malware-finder, accessed on 2021/7/23.

[16] D 盾, http://www.d99net.net/, accessed on 2021/7/23.

[17] WEBDIR+, https://scanner.baidu.com/#/pages/intro , accessed on 2021/8/2.