楊欣哲、黃妤萱（2021），『於雲端平台上設計 XML-based 包裹式攻擊防禦機制之研究』，*資訊管理學報*，第二十八卷，第二期，頁 155-182。

於雲端平台上設計 XML-based 包裹式攻擊防禦機制之研究

楊欣哲*
東吳大學資訊管理學系

黃妤萱
東吳大學資訊管理學系

摘要

由於雲端運算技術的迅速發展與應用普及，它是基於共享的計算資源透過虛擬化而形成的多台虛擬機器。政府企業皆使用雲端運技術來提高組織在資訊服務上的競爭力。雲端運算其服務可分為 IaaS、PaaS 與 SaaS 等三層，一般民眾也受益於雲端運算技術所帶來的好處。面臨雲端時代，雲端安全及用戶隱私也成為一項重要的議題。由於雲端用戶經常透過網路瀏覽器向雲端服務供應商請求服務，當一個經過簽署的訊息請求從服務供應端發送至服務接收端時，攻擊者可透過包裹式攻擊（Wrapping Attacks）來竄改網路上傳輸的 XML 或 SOAP 訊息，藉由躲避合法的驗證並在未被檢測到的情況下存取 Web 服務以實行包裹式攻擊。本文針對雲端運算環境下包裹式攻擊進行探討，透過整合並改善 Node Counting 預防機制，提出新的包裹式攻擊防禦機制，稱為 ENC-WRAP。

透過此 ENC-WRAP 防禦方法，延續 Node Counting 方法將判斷條件加以改良，分為攔截、檢測與記錄三個模組對傳入的 XML 或 SOAP 請求進行分析，除了比對子節點出現的次數，也對根節點至最終節點路徑上的元素進行檢測，以強化位在檢測模組上的驗證流程。模擬實驗結果顯示包裹式攻擊的 ENC-WRAP 方法相較 Node Counting 方法，於 50、100、200、500 個請求封包數量下，偵測率分別提升了 2%、3.8%、3.7%、2.8%，準確率分別提升 8%、7%、10.5%及 9.5%。由於將路徑上的元素納入考量，平均處理時間上分別多花費了 1%、1.3%、0.2%、0.6%的時間。未來，ENC-WRAP 方法可利用 Docker 虛擬機器技術改善計算資源之使用率以縮短平均處理時間。整體而言，本文所提出的 ENC-WRAP 方法，於雲端運算環境下能更準確地判斷包裹式攻擊者，進而降低雲端服務之安全風險並

* 本文通訊作者。電子郵件信箱：sjyang@csim.scu.edu.tw
2021/1/13 投稿；2021/2/5 修訂；2021/3/6 接受

提升其服務品質。

關鍵詞：雲端運算、ENC-WRAP、Node Counting、包裹式攻擊、SOAP

Yang, S.J. and Huang, Y.H. (2021), 'Design issues of XML-based wrapping attacks protection scheme in cloud platform', *Journal of Information Management*, Vol. 28, No. 2, pp. 155-182.

Design Issues of XML-based Wrapping Attacks Protection Scheme in Cloud Platform

Shin-Jer Yang*
Department of Computer Science and Information Management, Soochow University

Yu-Hsuan Huang
Department of Computer Science and Information Management, Soochow University

Abstract

Purpose－Cloud computing technology is a virtual machine formed through virtualization based on the sharing of computing resources. There are three service models including IaaS, PaaS and SaaS. Facing the cloud era, cloud security and user privacy have also become essential issues in cloud platform. Since cloud users often use Web browsers to request for services from cloud service providers, when a signed message request is sent to the service receiver from the service provider, attackers can use wrapping attacks to tamper with the XML or SOAP messages transferred through the internet in order to avoid legal verifications and access Web services without being detected to implement wrapping attacks. This paper is to combine and improve Node Counting mechanism to propose a new XML-based wrapping attack protection scheme called ENC-WRAP under cloud platform.

Design/methodology/approach－This ENC-WRAP is a continuation on the Node Counting method and the determining conditions were improved and divided into three modules: interception, detection and logging; it performs analysis to the incoming XML or SOAP requests. Not only does it compare the number of times the child node appears, but also perform detection to the elements on the path between the root node to the final node in order to enhance the verification procedure on the detection module.

Findings－The experimental results of KPIs for ENC-WRAP scheme indicate that the Detection rate can be increased by 2%, 3.8%, 3.7% and 2.8%; the Accuracy rate

* Corresponding author. Email: sjyang@csim.scu.edu.tw
2021/1/13 received; 2021/2/5 revised; 2021/3/6 accepted

can be increased by 8%, 7%, 10.5% and 9.5%; and the Average processing time is increased by 1%, 1.3%, 0.2% and 0.6% in 50, 100, 200, and 500 packet requests, respectively, under cloud computing environment. However, we will utilize Docker virtual machines technique to shorten the Average processing time.

Research limitation/implications ─ In the future, we will enhance the utilizations of computing resources and shorten the Average processing time by Docker virtual technology. Also, we will perform more simulations to consider the other KPI of APT (Average Processing Time) to obtain more efficient and effective results.

Practical implications ─ Practically, this paper designed new XML-based Wrapping Attacks Protection: scheme called ENC-WRAP to improve the verification process of the original Node Counting mechanism, the proposed ENC-WRAP can determine wrapping attackers more accurately to reduce the security risk and improve the quality of services in cloud computing platform.

Originality/value ─ The XML or SOAP request sent by the sender, which will be intercepted; as compared from Node Counting, not only is the frequency of child node appearances compared, elements on the path from the root node to the end node were also detected to discover illegal incoming requests and deny the requests for enhancing the verification process in proposed ENC-WRAP scheme.

Keywords: Cloud Computing, ENC-WRAP, Node Counting, Wrapping Attacks, SOAP

壹、緒論

一、研究背景

由於雲端運算技術的迅速發展與應用普及，它是基於共享的計算資源透過虛擬化而形成的多台虛擬機器。政府企業皆使用雲端運技術來提高組織在資訊服務上的競爭力。在雲端運算時代，相較於電腦發展初期將軟體安裝於個人電腦中，越來越多的軟體與資訊服務都能透過網路來取得。然而雲端運算並非憑空而生，之所以能有今日的雲端運算，是資訊產業歷經數十年的發展，奠基於包括虛擬化、網格運算、分散式運算、網際網路、SOA（Service-Oriented Architecture）、Web 2.0 等等的關鍵技術演化而成。雲端運算即為將網路、儲存、計算，或軟硬體及平台等 IT 資源，透過將其虛擬化與資源利用最佳化及可量化計費的服務型態，藉由網路分送，使得使用者可隨時取用的一種服務平台。隨著雲端發展漸趨成熟，有越來越多的企業將其服務與資料移轉至雲端來運作，以降低成本、改善內部業務流程、提高客戶的需求，藉此提升企業的價值。伴隨雲端技術的普及，不只企業透過雲端以提升其競爭優勢，一般民眾也受益於廣泛的雲端應用所帶來的好處，像透過將照片上傳至 Instagram，即可即時掌握朋友們的動態，或 Google 所提供的各樣服務：Gmail、Google 日曆、Google Docs 等等，讓生活更便利與豐富。而面臨雲端時代，雲端安全及用戶隱私也成為一項重要的議題。隨著雲端服務在網際網路上使用的增加，用戶經常透過網路瀏覽器向雲端服務供應商請求服務，服務通訊傳送過程中存在的敏感訊息、重要資料等內容容易引起惡意人員或駭客的入侵與利用。當 Web 伺服器驗證已簽署的請求，包裹式攻擊能夠在合法用戶與 Web 伺服器之間的 XML 或 SOAP 訊息轉換過程中完成。透過在登入時複製用戶的帳號及密碼，駭客將一個偽造元素（包裹器）嵌入到訊息結構中，在包裹器中挪動原始的訊息內文，轉而用惡意碼將其取代，然後將訊息發送到伺服器。由於原始內容仍然有效，因此伺服器將被欺騙，以授權實際上已被更改的訊息。因此，駭客能夠在受保護的資源上獲得未經授權的存取並執行蓄意的操作。

近期 Gupta 與 Santhi（2016）所提出的包裹式攻擊防禦方法（Node Counting），分為攔截、檢測與紀錄三個模組對傳入的 SOAP 請求進行分析，透過計算 SOAP Header 元素中子節點出現的次數來評估是否為包裹式攻擊，但當攻擊行為發生輕微變化或出現新的攻擊時，可能導致偵測率降低；換句話說，如果攻擊者設法改變訊息結構，可以很容易地繞過此防禦機制。因此本文將針對上述的 Node Counting 方法之缺失加以修正，提出一個明確指出 SOAP Header 或 XML 文件根節點到最終節點路徑上的每個元素來提高防禦能力的方法，稱為 XML-based 包裹式攻擊防禦方法，稱之為 ENC-WRAP。

二、研究目的

伴隨雲端運算技術的蓬勃發展，雲端上的安全性也成為值得探討的議題，由於雲端用戶經常透過網路瀏覽器向雲端服務供應商請求服務，因此，包裹式攻擊也會對雲端系統造成損害。透過修改授權的數位簽章的 XML 或 SOAP 訊息，駭客能夠獲得對受害者未經授權的存取。可以透過適當的身分驗證、將整個訊息做加密，以及與任何中間伺服器建立信任關係來防止包裹式攻擊的發生。本文所提出的 ENC-WRAP 將延續先前 Node Counting 防禦機制上的研究，進一步考量 SOAP Header 或 XML 文件根節點到最終節點上元素的路徑。總括，本論文主要的研究目的如下：

1. 雲端運算環境下，包裹式攻擊是一較新的威脅，本文延伸 Gupta 與 Santhi（2016）Node Counting 的包裹式攻擊防禦機制，提出新的 ENC-WRAP 方法改善此機制的驗證流程並提高雲端運算環境之安全性。
2. 針對由發送端傳入的 XML 或 SOAP 請求，首先將其攔截，透過檢測其屬性值且將簽署請求的絕對路徑納入考量以判斷是否為惡意請求並拒絕該請求，增加準確判斷之結果。
3. 本文所提出的 ENC-WRAP 方法，是透過 Burp Suite 與 SoapUI 模擬工具模擬雲端運算之實際環境，並與 Gupta 與 Santhi（2016）所提之預防機制做分析比較。於雲端環境下透過 ENC-WRAP 更有效地判斷包裹式攻擊者，所提供的服務能降低並減緩包裹式攻擊的發生。
4. 提出偵測率、準確率、平均處理時間等三項關鍵績效指標，與 Node Counting 防機制作分析比較，整體上，ENC-WRAP 提供更安全的雲端環境並提升其雲端服務品質。

三、章節結構

本論文架構於第壹節緒論中說明研究背景與動機以及研究範圍與目的。第貳節為文獻探討，探討目前包裹式攻擊之相關研究。第參節為研究架構方法，改良 Node Counting 方法以提出 ENC-WRAP 防禦，並說明其運作原理及設計其演算法，第肆節為模擬環境建置與績效指標說明，第伍節為模擬實驗與結果比較分析、第陸節為結論，說明本論文研究成果及未來的研究方向。

貳、文獻探討及相關研究

一、雲端運算及安全

美國國家標準暨技術研究院（National Institute of Standards and Technology; NIST）文件中，將雲端運算的定義為「雲端運算是一種模式，方便讓使用者能隨時隨地存取廣大的共享運算資源（如：網路、伺服器、儲存、應用程式和服務等），並可透過簡化工作的管理量以及與服務供應商的互動，快速提供各項服務。」（Mell et al. 2011）。在 NIST 定義中，雲端模型包含五種基本特徵、三種服務型式、四種佈署模式，如圖 1 所示。雲端運算之五種基本特徵包含以下幾點：

1. 隨需自助服務（On-demand Self-service）：使用者可以根據自身需求自行調整所需的資源多寡，像是服務時間及使用量，無需再由雲端服務供應商介入來調整。
2. 廣泛的網路存取（Broad Network Access）：網路使用無所不在，雲端供應商服務可隨時在網路上取用，各類型的使用端設備（例如：手機、平板、筆記型電腦、桌上型電腦等）均可透過標準機制連結至網路來使用服務。
3. 資源池共享（Resource Pooling）：供應商的運算資源被集中起來，依據使用者需求動態配置和重新指派不同的實體和虛擬資源，達到多位使用者共用的多租戶型式。其中資源包括儲存空間、記憶體、網路頻寬與虛擬伺服器等。
4. 快速且彈性（Rapid Elasticity）：能因應使用者需求快速調整資源的規模大小，對使用者而言，雲端供應商所提供的這項功能似乎是無限的，不受時間和資源規模的限制進行調配。
5. 可量化的服務（Measured Service）：提供的資源必須是能夠測量的，以利雲端供應商進行計費、存取控制、資源規劃等工作。

雲端運算之佈署模式包含以下四種：

1. 私有雲（Private Cloud）：針對單一組織來使用，通常為企業或組織為了自己而建立的雲端運算架構，避免會有因不同使用者而造成安全上的疑慮。其中，組織的資料和程式皆能透過私有雲在內部管理，可以彈性的運用且改善安全上的問題。
2. 公有雲（Public Cloud）：相較於私有雲，公有雲則是針對一般大眾提供的運算服務，但並不代表完全的開放，擁有者可對其存取控制進行設定，以防止有心人士查看公有雲上使用者的資料，也可以對使用者依據使用量或時間進行收費。一般常見的運算服務，大多數是屬於公有雲，像是針對個人的 Evernote、Dropbox、Google Docs，或是企業取向的 Microsoft Azure

等等。

3. 社群雲（Community Cloud）：社群雲由多個組織來共同成立，而這些組織的目的和利益通常是相近的，以服務擁有共同需求與訴求的群體，像是目前較廣為使用的教育雲、醫療雲等，藉此完成特定的任務或政策，社群雲可佈署於組織內或組織外部。
4. 混合雲（Hybrid Cloud）：混合雲也就是指由兩種或兩種以上不同雲端型態（私有雲、公有雲或社群雲）所組成的雲端架構，雲與雲之間雖然獨立存在，但兩者之間可以藉由專業技術連接在一起，以利資料和程式的轉移，例如，大型組織在處理資料方面可以使用公有雲的服務來運行，但若遇到較為機密的資料時，可由內部的私有雲來處理，以保障資料的安全。

雲端運算之服務模式主要有以下三種：

1. 軟體即服務（Software as a Service; SaaS）：軟體即服務為軟體的集合，這些應用程式架構於基礎架構層提供的資源及平台層提供的環境之上，並透過網路提供用戶使用。SaaS 提供的服務讓使用者不須將軟體下載至個人電腦，在不占用硬體資源的情況下，透過網際網路即可直接使用，如將 Office 軟體改以 Google Doc 方式線上作業，及同為 Google 提供的各樣服務如 Google Map、Gmail 等等。由於資料都儲存於雲端，對於供應商來說，能夠方便進行軟體的更新及佈署，使用者也可以降低管理和硬體維護等費用。
2. 平台即服務（Platform as a Service; PaaS）：PaaS 平台即服務主要針對軟體開發者提供完整的雲端開發環境，開發人員能夠透過此平台的開發工具將自己的應用程式直接佈署到雲端環境中進行測試，無須花心思維護平台背後的雲端架構，像是網路、伺服器、作業系統或儲存空間，僅須管理本身的應用程式和環境的設定，大幅降低以往自行建置平台及管理開發工具的費用及成本。PaaS 現有的服務提供商並不多，主要有 Microsoft 的 Microsoft Azure、Google 的 Google App Engine、IBM 的 Rational 等等。
3. 基礎設施即服務（Infrastructure as a Service; IaaS）：由雲端供應商提供基礎設施，像是伺服器、網路、儲存設備等，服務對象主要為 IT 管理人員，使用者可以依其自身業務需求彈性地租用所需的開發環境、自由選擇作業系統及應用程式，使用者不需要理解雲端運算背後的架構，只需將組態設定好便可取得供應商提供的硬體資源，如 Amazon 的 EC2 與中華電信的 HiCloud 即屬於 IaaS 服務。

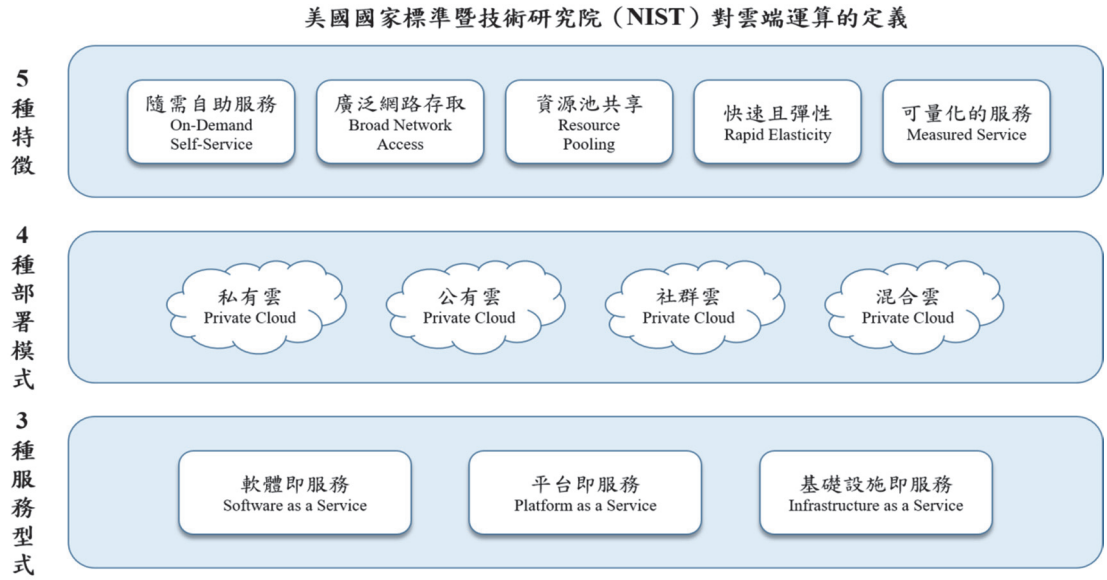

圖 1：雲端運算定義

一旦雲端運算時代來臨，雲端運算在資訊安全上是不論個人或企業皆必須面對的課題，近年雲端安全聯盟（Cloud Security Alliance; CSA, 2019）提出雲端運算會面臨的九大安全威脅，包含了以下幾點：資料外洩、資料遺失、帳號竊取、不安全的介面、阻斷服務攻擊、內部人員的惡意入侵、濫用雲端服務、貿然行動、共享技術所造成的安全議題。

對於雲端是否足夠安全，存在很多問題。隨著用戶經常透過網路瀏覽器向雲端服務供應商請求服務，服務通訊傳送過程中存在的敏感訊息、重要資料等內容容易引起惡意人員或駭客的入侵與利用。考慮到惡意入侵者，可能出現的攻擊有很多種，像是包裹式攻擊，可透過在登入階段複製使用者的帳戶與密碼，入侵在 Web 瀏覽器和伺服器之間交換的 XML 或 SOAP 訊息來完成。

二、XML Signature

XML Signature（XML 簽章）是一個定義數位簽章的 XML 語法的 W3C 推薦標準。從功能上，XML Signature 與 PKCS#7 有很多共同點，但是 XML 簽章具有更好的可擴展性，並為簽署 XML 文件做了調整。XML Signature 在許多 Web 技術中使用，如 SOAP、SAML 等等[1]。

XML signature 可以用來簽署任何類型的資源，最常見的是 XML 文件，但任

[1] https://zh.wikipedia.org/wiki/XML_Signature, 2019

何可以透過 URL 存取的資源都可以被簽署。

如果 XML 簽章用於對包含該簽章的 XML 文件之外的資源進行簽署，則稱為 detached signature；如果 XML 簽章用於對包含它的 XML 文件的某個部分進行簽署，則稱為 enveloped signature；如果 XML 簽章包含被簽署的數據，則稱為 enveloping signature[2]。

一個 XML 簽章包含一個 Signature 元素，其基本結構如圖 2 所示：

```
<Signature>
    <SignedInfo>
        <SignatureMethod/>
        <CanonicalizationMethod/>
        <Reference URI>
            <Transforms>
            <DigestMethod>
            <DigestValue>
        </Reference>
    </SignedInfo>
    <SignatureValue/>
    <KeyInfo/>
    <Object/>
</Signature>
```

圖 2：Signature 架構

三、SOAP 訊息

SOAP（Simple Object Access Protocol，簡單物件存取協定）是一種基於 XML 的訊息傳遞協定，使用描述 Web 服務發布的 XML 格式「WSDL」與基於 XML 的跨平台的描述規範「UDDI」(Kumar et al. 2017)，用於在電腦網路的 Web 服務中交換結構化訊息（McIntosh et al. 2005）。SOAP 仰賴於應用層協定，像是 HTTP 或 SMTP 等通訊協定，允許在不同作業系統上運作的程序以 XML 訊息格式進行資料交換與傳遞。

四、包裹式攻擊方式

簽章包裹式攻擊（Wrapping Attack）是一種利用惡意的中間節點引起的特定種類的攻擊，它透過添加一個新的元素干擾並操縱網路上傳輸的 SOAP 訊息。假定 Web 服務發送端發送一個經過簽署的訊息給服務接收端，從理論上講，服務接

[2] https://zh.wikipedia.org/wiki/XML_Signature, 2019

收端能夠檢測到經過簽署的訊息是否被惡意篡改，除非攻擊者能夠攻破服務發送端的簽章演算法。然而，攻擊者能夠通過躲避合法的驗證並在未被檢測到的情況下存取 Web 服務並施行簽章包裹式攻擊（李根來等 2011）。

XML 簽章包裹式攻擊由 McIntosh 及 Austel 在 2005 年首次提出（McIntosh et al. 2005），提出了以下四種包裹式攻擊的類型：

1. 簡單原始內文攻擊（Simple Ancestry Context Attack）：在簡單原始上下文攻擊中，請求的 SOAP 主體由簽章簽署，該簽章位於請求的安全標頭中。訊息的接收者檢查簽章是否正確，並將簽章憑證中的信任合法化。最後，接收者透過在簽章中帶入 SOAP 主體的“id”到 ID 參考來了解所需的元素是否實際被簽署（McIntosh et al. 2005）。
2. 可選元素內文攻擊（Optional Element Context Attack）：在可選要素上下文攻擊中，簽署的數據包含在 SOAP 標頭中，並且是隨機的。將這種攻擊與先前的簡單內文攻擊（Simple Context Attack）相比，主要問題並不是 SOAP 標頭中簽署數據的位置。事實上，其簽章數據的可選擇性才為主要的問題。
3. 同輩值內文（Sibling Value Context）：同輩值內文攻擊中，安全標頭包含一個簽署的元素，這實際上是<Signature>的替代同輩值。這種攻擊的常見模型可以是<Timestamp>的元素，它與<Signature>同樣是 SOAP 安全標頭的直屬後裔，此攻擊和先前提及的攻擊的區別在於簽署的數據－在此次攻擊中為<Signature>的同輩值（McIntosh et al. 2005）。這種攻擊的主要目的是忽略簽章元素的同輩值。
4. 同輩排序內文（Sibling Order Context）：根據 McIntoch 和 Austel 於 2005 所提出，這種攻擊是為了保護個別簽署的同輩元素。它們的語義與它們相對於彼此的順序有關，不受攻擊者重新排序的影響。需要更多的研究來定義不妨礙同輩值的增刪修改且不影響其語意排序的適當對策。

為了確保交換的 XML 文件的完整性和真實性，XML 安全工作小組定義了 XML 簽章規範。XML 簽章允許將密碼原始地應用於 XML 訊息，從而保護任意 XML 元素。根據 WS-Security 標準，應用於 SOAP 訊息的 XML 簽章的簡化結構如圖 3 所示。

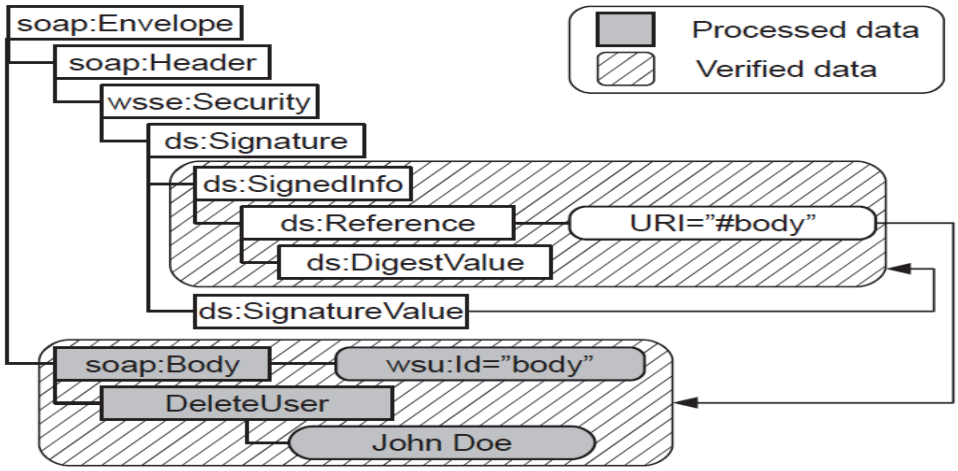

圖 3：在 SOAP 主體上應用的簽章範例

圖 3 描繪的 SOAP 訊息包括刪除用戶“John Doe”的函數調用－其在 SOAP 訊息的主體中定義。 SOAP 標頭中定義的 XML 簽章確保了 SOAP 正文的真實性和完整性。 XML 簽章元素由兩個必需元素組成：<SignedInfo>和<SignatureValue>。 <SignedInfo>元素包含一個指向 SOAP 正文的 Id-reference 以及透過引用的元素計算出的摘要值。 為了保護<SignedInfo>元素，計算該元素上的簽章值並將其放入<SignatureValue>元素中。 這通常透過諸如 RSA 或 DSA 的公開金鑰演算法完成。

以下說明給定 SOAP 訊息的處理：收件人首先搜索<SignedInfo>中給出的被引用元素。接著計算該元素的摘要值並將其與<DigestValue>元素中給出的值進行比較。之後，藉由<SignedInfo>驗證簽章值。在最後，它可以執行 SOAP 主體中定義的函數。可以看出，處理 SOAP 訊息和驗證包含的 XML 簽章有兩個獨立的步驟。首先由 McIntosh 和 Austel 提出了這一觀察結果，他們濫用不同的處理位置進行新的攻擊，也就是 XML 簽章包裝（XML Signature Wrapping）。圖 4 提出了一個 XML 簽章包裹式攻擊的例子。

在這個例子中，竊聽訊息的攻擊者將原始 SOAP 主體移動到 SOAP 標頭。 之後，他創建了一個新的 Id="attack" 的 SOAP 主體，並定義了一個任意函數：在這個例子中，強制商業邏輯執行一個賦予“John Doe”管理權限的函數。 由於原始 SOAP 主體的 Id 保持不變並且相關部分沒有被改變，所以安全邏輯可以驗證其完整性和真實性。商業邏輯也將新定義的 SOAP 主體作為輸入。

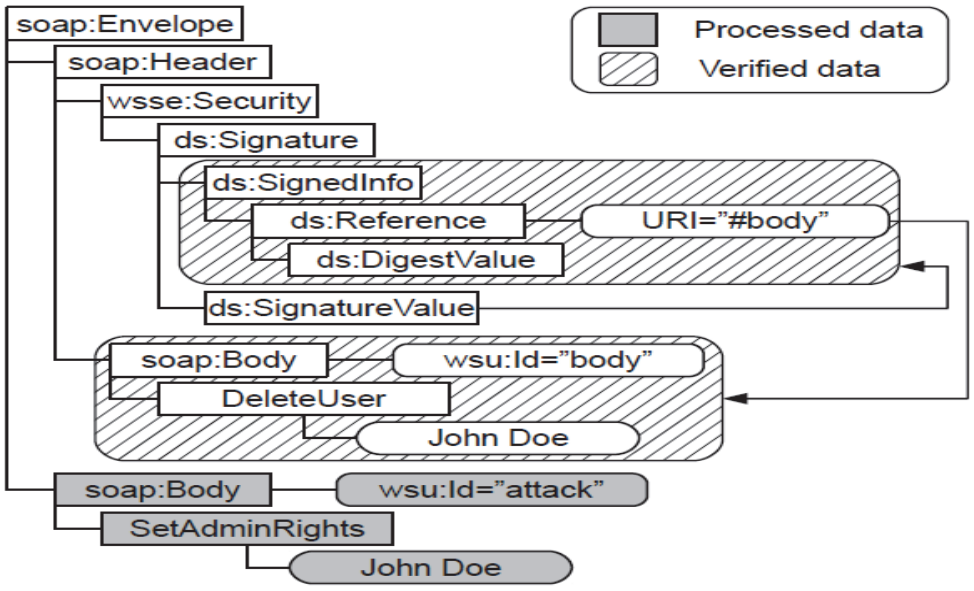

圖 4：簽章包裹式攻擊

XML 包裹式攻擊是一項新發現的攻擊，如前面所解釋，這類型的攻擊利用了處理 XML 簽章時產生的漏洞，藉由在請求中注入惡意資料來成功實行包裹式攻擊。在下小節我們將討論針對包裹式攻擊現有的防禦方法。

五、包裹式攻擊防禦機制與相關應用

在現實世界中，由於雲端用戶通常透過網路瀏覽器向雲端服務供應商請求服務，因此包裹式攻擊也會對雲端系統造成損害，亞馬遜的 EC2 在 2008 年被發現容易受到這類型的攻擊（Somorovsky et al. 2011），研究指出 EC2 在 SOAP 訊息安全驗證機制有其弱點，駭客可藉由攔截及修改合法用戶的簽署 SOAP 請求，對雲端中的受害者帳戶採取未經授權的行動。在（Somorovsky et al. 2011）中，提出一稱為「黑箱」的分析方法，針對亞馬遜 EC2 上的公有雲平台來做研究。

目前與 XML Signature 包裹式攻擊相關的研究並不多，XML 簽章包裹式攻擊利用了處理 XML 簽章時所產生的漏洞，在訊息傳送的過程中，攻擊者使用偽造元素將原始元素替換，或將 SOAP 訊息中的原始元素從原始位置重新定位，藉此於請求中注入惡意資料。XML 包裹式攻擊由 McIntosh 及 Austel 在 2005 年首次提出（McIntosh et al. 2005），並對如何防範包裹式攻擊做初步探討，藉由在發送方與接收方兩端制定安全策略規範預防簽章對非同意集合成員的元素做參考，從而拒絕包含非預期簽署元素的訊息。Saudi 等（2019）提出 SSM（Spatial Signature

Method）的分析方法，以防範 XML 簽章包裹式攻擊。上述這些方法雖然在某些情況下尚令稱滿意，但這些方法並非在所有情況下皆為適用。

近年來，有關包裹式攻擊防禦機制有 Schema Hardening、SESoap、FastXPath、Node Counting 四種，每一機制分別說明如下：

1. Schema Hardening：Schema Hardening 允許來自規範自身描述的與原始 Schema 相同的有效 XML 文件的集合，但還需要強化以嚴格禁止 Schema 中未包含的任何其他內容，它涉及從規範本身刪除原始 Schema 文件中出現的所有 Schema 擴展點的技術（Jensen et al. 2011）。確切來說，Schema 強化調查了每一個出現的<xs:any>宣告，驗證是否用於普遍的 SOAP 訊息，將之重構或刪除。但強化的 Shema 應用也帶來其顯著的性能劣勢。
2. SESoap：SESoap 方法即簽署整個 SOAP 方法，它是將數位簽章結構應用於整個 SOAP 封裝元素，從而保障整個文件的安全性（Kouchaksaraei et al. 2013）。因此，攻擊者將無法更改元素的位置或將任意元素新增、刪除到原始文件中。
3. FastXPath：FastXPath 方法要求明確指出從文件根目錄到簽署子樹路徑上的每個元素，使得沒有彈性能將任意被簽署的內容移動至文件中的其他位置（Gajek et al. 2009）。此外，由於 FastXPath 陳述式可包含任何的 ID 屬性，因此它安全地支持了 ID-based 與 structure-based 的兩種 XML 存取方法，因為這兩種方法都指向相同的元素，所以簽章驗證函數的存取方法及應用程式邏輯的存取方法間不會出現偏差。然而，當存取規則運用於不同的程式邏輯中，包裹式攻擊的威脅也無法完全避免。
4. Node Counting：Node Counting 為近期提出的包裹式攻擊防禦機制，其系統設計分為攔截、檢測和記錄三個主要模組（Gupta & Santhi 2016）。第一個攔截模組負責攔截傳入的數位簽署 SOAP 請求並將其轉發給檢測模組。接著檢測模組運用演算法透過分析請求屬性與計算子節點出現的頻率來檢測包裹式攻擊是否存在，如果檢測到，則拒絕該請求，並透過記錄模組生成記錄（Log）；若未檢測到包裹式攻擊，則將請求轉發給目標接收者，並產生成功轉發請求的另一個記錄。

在上述四種方法中，Schema Hardening 與 SESoap 兩種方法其處理過程較為複雜且嚴謹，進而保障整個 XML 文件的安全性，因此花費較多的處理時間。FastXPath 方法與近期提出的 Node Counting 方法之處理過程則較前兩者彈性許多，改善處理時間花費過多的問題。Node Counting 為上述四種方法中最新提出的防禦機制，其優點為計算上負荷較少，並且在保護 SOAP 訊息交換上有良好的效能，然而，當結構出現輕微變化時，可能降低其方法的效能。總之，此四種方法特徵對照比較如表 1 所示。

表 1：包裹式攻擊防禦方法特徵比較

行為 / 方法	屬性驗證	刪除所有擴展點	將簽章架構套用於整個 SOAP	指出路徑上的每個元素	計算節點次數
Schema Hardening (2011)	V	V			
SESoap (2013)	V		V		
FastXPath (2009)	V			V	
Node Counting (2016)	V				V

參、ENC-WRAP 之運作原理與演算法設計

一、ENC-WRAP 運作原理與流程

本研究延續近期所提出較新的 Node Counting 方法針對其檢測上的缺失加以改良，提出 XML-based 包裹式攻擊防禦機制 ENC-WRAP 方法，結合 SoapUI 與 Burp Suite Professional 兩套模擬工具分為三個模組進行偵測，三個模組分別為攔截模組、檢測模組與紀錄模組。第一階段首先由 SoapUI 模擬工具傳送合法與非法的 XML 或 SOAP 請求，接著在 Burp Suite 滲透測試工具上的 Proxy 模組將傳入的數位簽署 XML 或 SOAP 請求攔截並轉送給檢測模組，第二階段檢測模組透過演算法徹底分析請求屬性，同時進行過濾以檢測是否存在包裹式攻擊，第三階段透過記錄模組生成記錄。ENC-WRAP 方法延續 Node Counting 方法針對其檢測上的缺失加以改良，首先透過 XPath 陳述式查看傳入的 XML 或 SOAP 簽署請求是否為<Header>的子元素，接續除了計算子節點出現的次數，也對文件根節點到最終節點路徑上的元素進行檢測，以找出非法的傳入請求。

本文的 ENC-WRAP 之詳細運作流程如圖 5 所示，其方法之主要的特色如下：

1. 能夠攔截由發送者傳入的 XML 或 SOAP 請求，並轉送給檢測模組，若檢測到為非法的請求，則拒絕該請求，並透過記錄模組生成記錄；若檢測到為合法請求，則將請求轉發給目標收件人，並於記錄模組記錄成功轉發請求。
2. 本文所提出包裹式攻擊防禦機制可適用於雲端服務的多種類型。
3. 除了比對 XML 或 SOAP Header 中子節點出現的次數，同時將 XML 或 SOAPSOAP Header 中根節點到最終節點路徑上每個元素納入考量，以提升包裹式攻擊防禦能力。

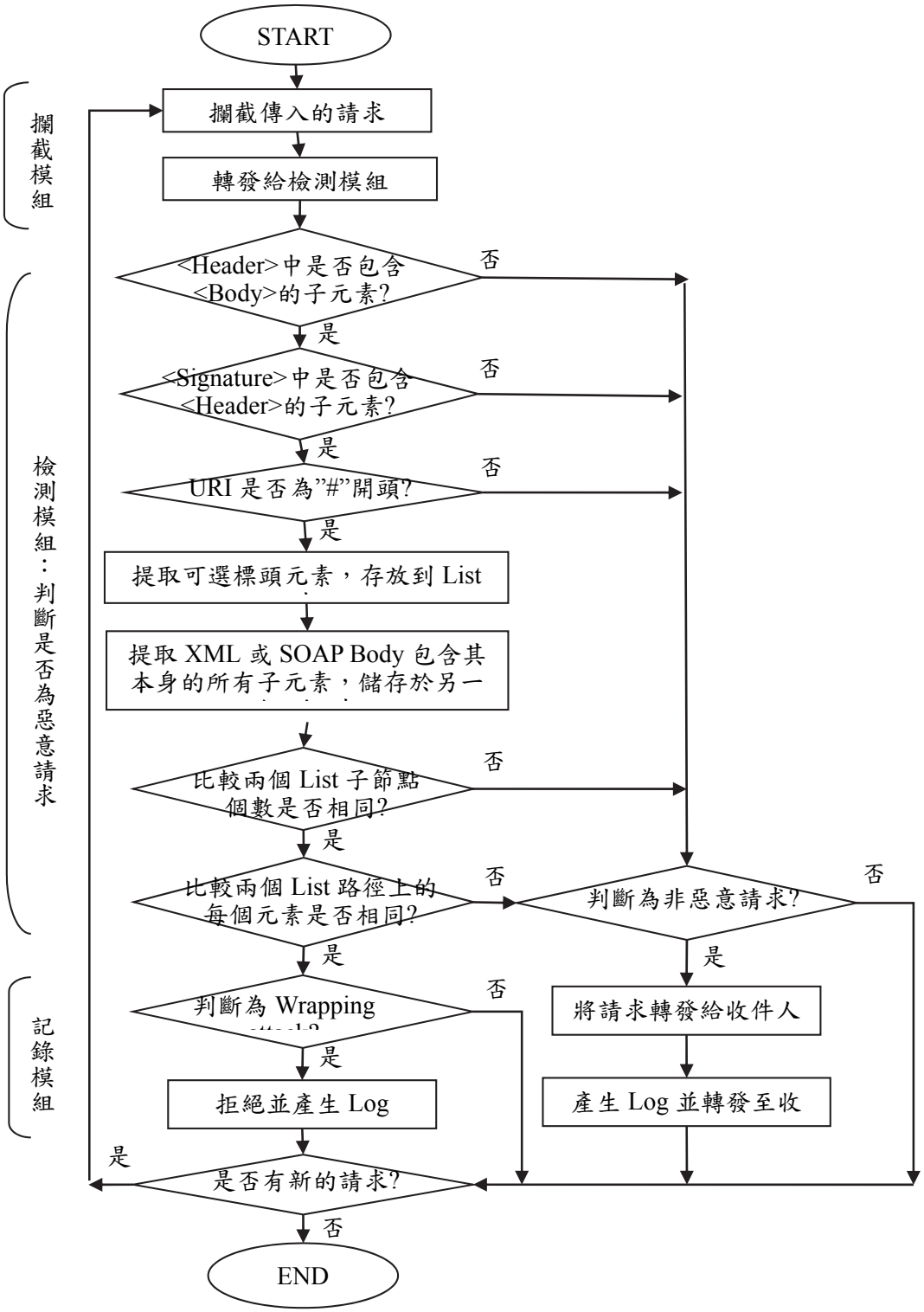

圖 5：ENC-WRAP 運作流程圖

二、ENC-WRAP 演算法設計

根據圖 5：ENC-WRAP 運作流程圖，設計出 ENC-WRAP 演算法。本文所提出的 ENC-WRAP 方法延續 Node Counting 方法針對其檢測上的缺失加以改良，新增判斷條件如判斷<Header>中是否包含<Body>子元素、<Signature>是否為<Header>的子元素以及判斷 URI 是否為”#”開頭，並對簽署請求的 URI 值進行檢驗。除了延續 Node Counting 方法運用 XPath 陳述式計算子節點出現的次數，也結合同樣運用 XPath 陳述式計算的 FastXPath 方法－比對文件根節點至最終節點路徑上的元素來加強檢測流程，進而提高對包裹式攻擊之防禦。總之，其 ENC-WRAP 演算法之程序虛擬碼說明如下：

```
Algorithm ENC-WRAP()
{
Input:
String[] ENC-WRAP;              //請求集合
bool boRequestInput;            //是否還有請求傳入
bool boENC-WRAP;                //判斷讀入的子元素是否為正確
bool boAttacker                 //是否為攻擊者
bool boFakeRequest              //是否為惡意之請求
Int[] List1                     //第一個 List 出現的子節點數
Int[] List2                     //第二個 List 出現的子節點數
String[] ListPath1              //儲存第一個 List 的節點
String[] ListPath2              //儲存第二個 List 的節點
String ENC-WRAP_Evil
String Attack
Output: To complete ENC-WRAP Method.
Method:
BEGIN
boRequestInput == true
{
While(boRequestInput) {         //若有新傳入請求則繼續
    ENC-WRAP_Load();            //讀入請求
    ENC-WRAP_Body();            //判斷<Header>是否包含<Body>的子元素
    If boENC-WRAP == true then
        ENC-WRAP_Sign();        //判斷<Signature>是否為<Header>的子元素
```

```
        If boENC-WRAP == true then
        ENC-WRAP_Begin();  //判斷 URI 是否為”#”開頭
        If boENC-WRAP == true then
        ENC-WRAP_Input();

    END If
}WEND
    List_Count();           //比較兩個 List 子節點個數是否相同
    List_Path();            //比較兩個 List 路徑上的每個元素是否相同
    boAttacker = ENC-WRAP_Check(); //分析請求，判斷是否為攻擊者
    if(boAttacker == true or boFakeRequest == true){  //判定為攻擊者
        If boENC-WRAP = false
            then List_Illegal_Request();        //列為非法請求
        }else{                                  //判定為非攻擊者
        List_Legal_Request();                   //列為合法請求
        }
    if(boRequestInput) == true then ENC-WRAP_Input(); //確認是否還有請求流入
    }
} END
Procedure ENC-WRAP_Load() {                     //讀入請求
    String ENC-WRAP = getENC-WRAP();
    ENC-WRAP = ENC-WRAP.Split( ‘ ’ );           //將請求拆解至請求集合
} END ENC-WRAP_Load
Procedure ENC-WRAP_Body() {     //判斷<Header>中是否包含<Body>的子元素
    if(head.contains(body)){
        return true;
    }else{
        return false;
    }
} END ENC-WRAP_Body
Procedure ENC-WRAP_Sign() {     //判斷<Signature>是否為<Header>的子元素
    if(head.contains(signature)){
        return true;
    }else{
        return false;
```

```
    }
} END ENC-WRAP_Sign
Procedure ENC-WRAP_Begin() {      //判斷 URI 是否為”#”開頭
    if(str.startsWith("#")){
        return true;
}else{
return false;
    }
} END ENC-WRAP_Begin
Procedure List_Count() {          //比較兩個 List 子節點個數是否相同
    if(List1.length==List2.length){
        return true;
}else{
return false;
}
} END List_Count
Procedure List_Path() {       //比較兩個 List 路徑上的每個元素是否相同
    if(ListPath1.equals(ListPath2)){
        return true;
    }else{
        return false;
    }
} END List_Path
Procedure ENC-WRAP_Check() {      //判斷是否為 Attack
if(ENC-WRAP_Evil=Attack){
        boFakeRequest == true
    return true;
    }else{
        return false;
    }
} END ENC-WRAP_Check
Procedure ENC-WRAP_Input() {      //檢查是否新的請求流入？
    if(boRequestInput){           //若有新的請求流入
        return true;
    }else{                        //若無新的請求流入
```

```
            return false;
        }
} END ENC-WRAP_Input
}
End ENC-WRAP.                    //結束 ENC-WRAP
```

肆、模擬環境建置與績效指標說明

一、模擬環境建置

本研究將透過虛擬機器與實體機器的方式模擬雲端的測試環境，將在一台實體電腦上安裝 Oracle VM Virtual Box 6.0.4 版虛擬機器軟體，在虛擬機器中安裝 Ubuntu 18.04 版本之 Linux 作業系統軟體，JDK 版本為 1.9.0。本研究將以第一台虛擬機器設定為雲端環境架構之主節點伺服器，再設定一台同樣建置雲端運算平台並可執行 ENC-WRAP 模擬器的虛擬機器，首先啟動 SoapUI 發送合法與非法的 XML 或 SOAP 請求，接著在 Burp Suite Professional v1.7.37 攔截傳入請求並轉送至位在 SoapUI 上的檢測模組，透過演算法分析請求屬性同時進行過濾以檢測是否存在包裹式攻擊。評估各項相關績效指標。模擬環境系統配置與實驗工具功能說明如表 2、表 3 所示。

表 2：模擬環境系統配置

軟、硬體配置	軟、硬體規格	虛擬機器
Operation System	Windows 10	Ubuntu 18.04
CPU	Intel® Core™ i5-7200U CPU @ 2.50GHz 2.7GHz	2 Cores 2.5 GHz
Memory	8 GB	2 GB
Disk	224 GB	40 GB

表 3：模擬實驗工具功能

模擬實驗工具	功能說明
SoapUI	SoapUI 是一個用於測試 XML 或 SOAP 與 REST API 的工具，涵蓋 Web 服務檢查、呼叫、開發、模擬、安全性測試、負載測試等功能。可於 SoapUI 工具中載入測試元件，並執行完整的弱點掃描。

Burp Suite Professional v1.7.37	Burp Suite 是一套用於檢測 Web 應用安全的滲透測試工具，集合了多種測試元件，可攔截伺服端與用戶端間往返的請求資訊，並對攔截的訊息進行各種處理。

本模擬實驗蒐集請求資料來源使用 WSDL URL 之 XML 或 SOAP 訊息請求做為測試（網址：http://webservices.oorsprong.org/websamples.countryinfo），將這些 XML 或 SOAP 訊息放入 Wrapper 改變其原始架構以實行包裹式攻擊。實驗中模擬包裹式攻擊者所發送的請求在雲端環境下傳遞的情況：當 XML 或 SOAP 請求由發送端傳送至接收端時，首先將其攔截，接著對 XML 或 SOAP 請求進行分析，結合 ENC-WRAP 檢測機制來確認 XML 或 SOAP 請求是否為惡意請求，若為惡意請求則拒絕該請求，模擬時會對包裹式攻擊者進行限制，其模擬環境架構示意如圖 6 所示。

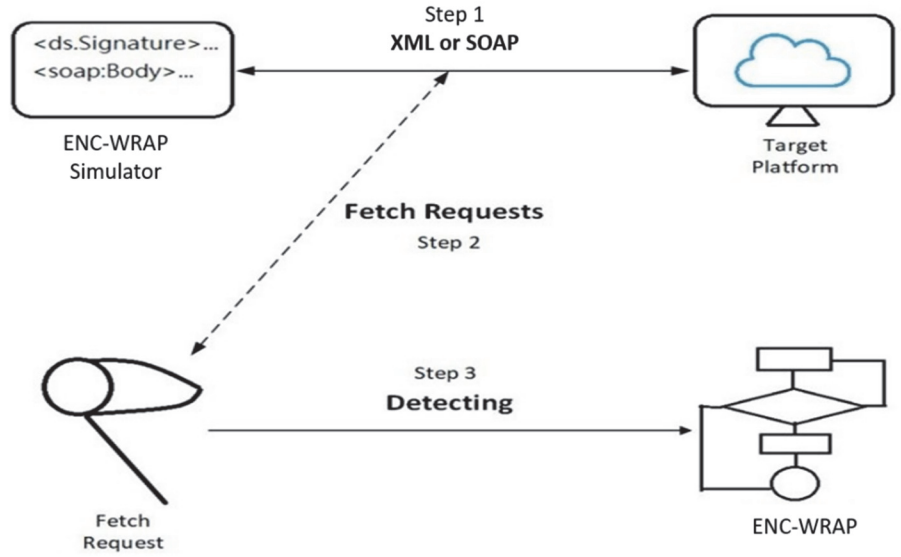

圖 6：模擬環境架構示意圖

二、模擬績效指標說明

在本文中，我們將觀察三種關鍵績效指標（KPIs）來分析模擬結果，分別為偵測率（Detection Rate）、準確率（Accuracy Rate）、平均處理時間（Average Processing Time），其詳細內容說明如表 4 所示以及關鍵績效指標計算如公式(1)、(2)、(3)。

表 4：關鍵績效指標定義說明

關鍵績效指標（KPIs）	定義及分析目的
偵測率 (Detection Rate; DR) Unit: %	計算被偵測出的異常請求數（DAR）佔總異常請求數（TAR）的比例，並與 Node Counting 方法對照比較。如公式(1)
準確率 (Accuracy Rate; AR) Unit: %	計算判斷正確請求數（CR）佔請求總數（N）的比例，藉此評估 ENC-WRAP 與 Node Counting 方法的準確性。如公式(2)
平均處理時間 (Average Processing Time; APT) Unit: ms	將總處理時間（TPT）除以請求總數 N，藉此得出每個請求的平均處理時間，以評估 ENC-WRAP 與 Node Counting 方法的處理效率。計算如公式(3)

$$DR(\%) = DAR/TAR \times 100 \quad (1)$$

$$AR(\%) = CR/N \times 100 \quad (2)$$

$$APT(ms) = TPT/N \quad (3)$$

本論文設計之 ENC-WRAP 演算法將採用 JAVA 語言開發，並將與 Node Counting 方法透過實驗進行模擬，針對提出如表 4 中三個關鍵績效指標（KPIs）做數據分析比較。詳細模擬程序說明如下。

1. 研究並設計 ENC-WRAP 之模擬實驗程序。
2. 將 ENC-WRAP 設定於雲端環境，使用 SoapUI 與 Burp Suite Professional v1.7.37 模擬工具進行檢測。
3. 設定好模擬環境及預設相關參數，執行並進行 ENC-WRAP 與 Node Counting 提出之方法之績效比較。
4. 蒐集經由多次模擬後的平均關鍵績效指標相關數據，以提高可信度。
5. 依照蒐集的數據進行實驗結果與績效分析。

伍、實驗結果與比較分析

本實驗蒐集請求資料來源使用 WSDL URL 之 XML 或 SOAP 請求做為測試，加入 Wrapper 改變其原始結構，並於 Burp Suite Professional 及 SoapUI 模擬工具上執行，首先從 SoapUI 模擬工具中發送請求，接著於 Proxy 模組中攔截由發送者傳入的 XML 或 SOAP 請求並轉送至 Scanner 模組，接續於 Scanner 模組上徹底分析請求進行過濾篩選，以找出非法的傳入請求，若偵測為非法請求則拒絕該請求，最後於紀錄模組上進行分析統計並生成實驗結果。

針對偵測率（Detection Rate）的實驗數據分析，本實驗攔截由發送者傳入的 XML 或 SOAP 請求，以不同請求數量情形下於檢測模組進行測試。模擬實驗結果顯示本文所提之 ENC-WRAP 方法在 50 個、100 個、200 個、500 個請求封包數量下，偵測率分別為 98%、98.8%、99.2%、98.5%，如圖 7 所示。

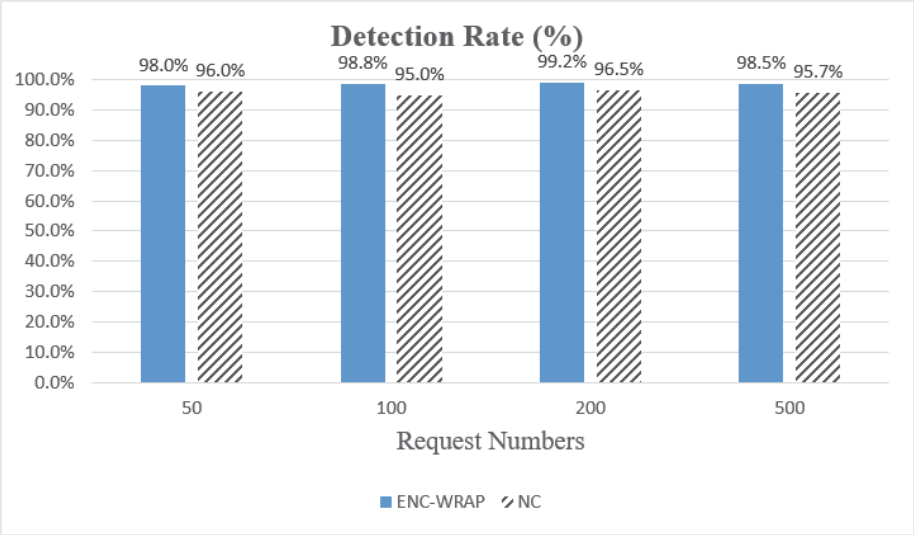

圖 7：ENC-WRAP 與 NC 於不同請求數的偵測率比較

針對準確率（Accuracy Rate）的實驗數據分析，本實驗攔截由發送者傳入的 XML 或 SOAP 請求，以不同請求數量下於檢測模組進行測試。由於 Node Counting 只考量子節點出現的頻率，在子節點數量相同的情形下，可能因結構或元素位置不同使得正常請求被判斷為非法請求，導致誤判的情形發生。實驗結果顯示本文所提 ENC-WRAP 方法在 50 個、100 個、200 個、500 個請求封包數量下，準確率分別為 98%、98%、98.5%、99.2%，比 Node Counting 方法獲得較高的準確率，如圖 8 所示。

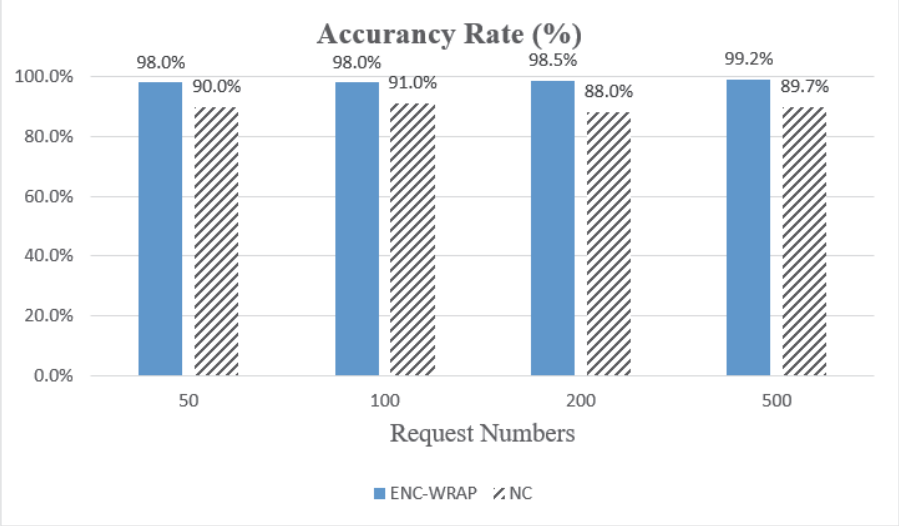

圖 8：ENC-WRAP 與 NC 於不同請求數的準確率比較

針對平均處理時間（Packet Latency）的實驗數據分析，本實驗攔截由發送者傳入的 XML 或 SOAP 請求，以不同請求數量下於檢測模組進行測試，由於 ENC-WRAP 需將 XML 或 SOAP Header 中路徑上的每個元素納入考量，因此平均處理時間相較 Node Counting 有微幅增加。模擬實驗結果顯示本文所提之 ENC-WRAP 方法在 50 個、100 個、200 個、500 個請求封包數量下，單一請求的平均處理時間分別為 662、644、635 及 631 毫秒，如圖 9 所示。

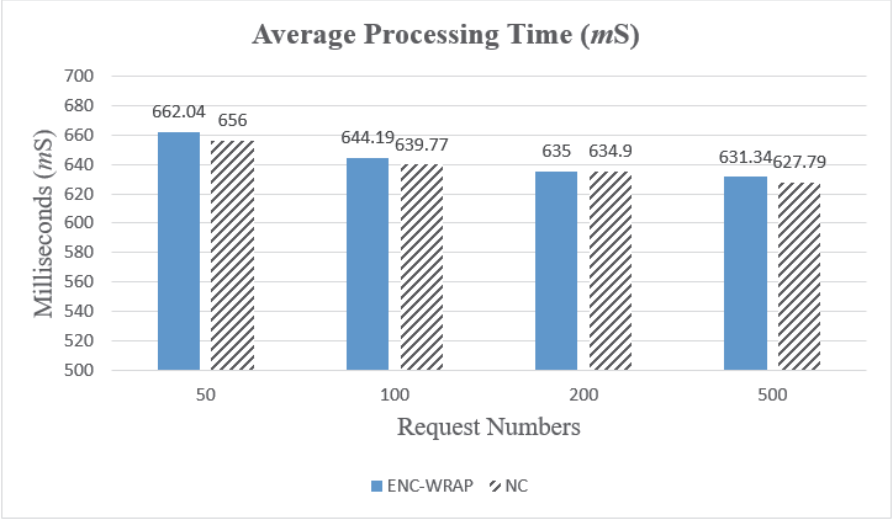

圖 9：ENC-WRAP 與 NC 於不同請求數的平均處理時間比較

總之，本研究依據 ENC-WRAP 方法及 Node Counting 方法的模擬實驗執行結果進行彙總，整體實驗結果如表 5 所示。ENC-WRAP 方法在請求封包數量為 50 個、100 個、200 個、500 個情形下，偵測率分別為 98%、98.8%、99.2%、98.5%，準確率分別為 98%、98%、98.5%、99.2%，平均處理時間為 662、644、635、631 毫秒。與 Gupta and Santhi（2016）所提出 Node Counting 方法相比，不同請求個數下的改善比率如表 6 所示。模擬執行結果，其偵測率分別提升了 2%、3.8%、3.7%、2.8%，準確率分別提升 8%、7%、10.5%及 9.5%，平均處理時間則分別增加 1%、1.3%、0.2%、0.6%。從結果看出，ENC-WRAP 方法在準確率與偵測率上較 Node Counting 表現得較好，由於 ENC-WRAP 偵測流程需將路徑上元素納入考量，因此平均處理時間較 Node Counting 方法有微幅增加，但整體而言，本研究所提出之 ENC-WRAP 方法能提供更安全的服務品質。

表 5：ENC-WRAP 與 Node Counting 模擬執行結果之彙總

ENC-WRAP				
請求封包總數 / KPIs	50	100	200	500
偵測率	98%	98.8%	99.2%	98.5%
準確率	98%	98%	98.5%	99.2%
平均處理時間	662	644	635	631
Node Counting				
請求封包總數 / KPIs	50	100	200	500
偵測率	96%	95%	96.5%	95.7%
準確率	90%	91%	88%	89.7%
平均處理時間	656	639	634	627

表 6：ENC-WRAP 相對於 Node Counting 不同請求數下的改善比率

請求封包總數 / KPIs	50	100	200	500
偵測率	2%	3.8%	3.7%	2.8%
準確率	8%	7%	10.5%	9.5%
平均處理時間	-1% (6*ms*)	-1.3% (5*ms*)	-0.2% (1*ms*)	-0.6% (4*ms*)

陸、結論

隨著用戶經常透過網路瀏覽器向雲端服務供應商請求服務，服務通訊傳送過程中存在的敏感訊息、重要資料等內容容易引起惡意人員或駭客的入侵與利用。攻擊者可利用包裹式攻擊（Wrapping Attack）在訊息傳輸過程中將惡意元件注入到合法的訊息結構，透過躲避合法驗證並在未被檢測到的情況下存取 Web 服務以實行包裹式攻擊。因此，本論文以 Gupta 與 Santhi（2016）的 Node Counting 防禦機制為基礎，提出一項 XML-based 包裹式攻擊之改進節點技術方法，稱之為 ENC-WRAP，以改善原本機制的驗證流程。

在 ENC-WRAP 防禦機制中，運用 SoapUI 與 Burp Suite Professional v1.7.37 兩套模擬工具，分為攔截、檢測與紀錄三個模組對傳入的 XML 或 SOAP 請求進行分析，針對由發送端傳入的 XML 或 SOAP 請求，首先將其攔截，延續 Node Counting 方法除了比對子節點出現的頻率外，也對根節點至最終節點路徑上的元素進行檢測，以找出非法的傳入請求並拒絕該請求，強化位在檢測模組上的驗證流程，結果顯示 ENC-WRAP 方法在請求封包數量為 50 個、100 個、200 個、500 個情形下，偵測率分別為 98%、98.8%、99.2%、98.5%，準確率分別為 98%、98%、98.5%、99.2%，平均處理時間為 662、644、635、631 毫秒。與 Gupta 與 Santhi（2016）所提出 Node Counting 方法相比，不同請求個數下，偵測率分別提升了 2%、3.8%、3.7%、2.8%，準確率分別提升 8%、7%、10.5%及 9.5%，平均處理時間則分別增加 1%、1.3%、0.2%、0.6%。由此得知，本文所提出的 ENC-WRAP 方法在偵測包裹式攻擊方面有較高的偵測率與準確率，而由於在偵測模組上將路徑上元素納入考量，因此在平均處理時間上有微幅增加，但整體而言，ENC-WRAP 方法相較於 Node Counting 方法能更準確地判斷包裹式攻擊者，並提升雲端服務之安全及服務品質。

本論文之整體研究成果及具體貢獻如下：

1. 本研究以 Gupta 與 Santhi（2016）的 Node Counting 預防機制為基礎，改善原本機制的驗證流程，提出一項改進的混合式節點技術方法，稱為 ENC-WRAP。
2. 針對由發送端傳入的 XML 或 SOAP 請求，首先將其攔截，除了延續 Node Counting 方法比對子節點出現的次數，同時檢測其屬性值並結合 FastXPath 方法將簽署請求的元素路徑納入考量以判斷是否為惡意請求並拒絕該請求，增加準確判斷之結果。
3. 以本文提出的 ENC-WRAP 方法，於 Burp Suite 與 SoapUI 模擬工具模擬 XML 或 SOAP 請求傳送之實際環境，並與 Gupta 與 Santhi（2016）所提之預防機制做分析比較，透過 ENC-WRAP 能更有效地判斷包裹式攻擊者。

總之，此包裹式防禦機制：ENC-WRAP 能於雲端平台上所提供的服務可降低並減緩包裹式攻擊的發生。

4. 本文提出三項關鍵績效指標，與 Node Counting 預防機制作分析比較，模擬結果顯示 ENC-WRAP 方法獲得更好的偵測率與準確率，但在計算上較複雜因而多花費一些處理時間，此部分可利用 Docker 虛擬技術以縮短平均處理時間。

整體而言，ENC-WRAP 能於雲端環境下提供更安全且有保障的雲端服務環境。隨著雲端運算的蓬勃發展，用戶經常透過瀏覽器向雲端服務供應商請求服務，包裹式攻擊能在請求傳送過程中完成，進而對受害者帳戶採取未經授權的行動，本文提出 ENC-WRAP 防禦機制，能將請求攔截並對請求進行檢測與分析，比對簽署請求的元素路徑與節點出現次數以判斷是否為包裹式攻擊，若判斷為惡意請求則拒絕該請求，減少包裹式攻擊發生的可能性，讓雲端用戶能獲得更有保障的雲端服務環境，提升用戶在雲端上的服務品質。

未來，我們將擴大模擬實驗執行，在屬性偵測上考量如何下參數值來獲取更為準確的分析，同時改進當節點數過多而增加處理時間的問題，並藉由利用 Docker 虛擬機器技術協助以縮短平均處理時間，並納入不同的 KPIs 如系統處理能量獲得更為客觀的實驗結果，以證明本文所提出 ENC-WRAP 方法能在雲端環境下針對包裹式攻擊提供更有效且完善的防禦機制。

誌謝

本論文承蒙兩位匿名審查委員之寶貴意見與悉心指正，使得本論文得已修正並更加完善，謹此致謝。

參考文獻

李根來、趙逢禹（2011），『基於策略斷言的 SOAP 消息簽名包裝攻擊檢測』，未出版碩士論文，上海理工大學，上海。

維基百科，『SOAP 簡單物件存取協定』，https://zh.wikipedia.org/wiki/%E7%AE%80%E5%8D%95%E5%AF%B9%E8%B1%A1%E8%AE%BF%E9%97%AE%E5%8D%8F%E8%AE%AE, Retrieved on 2019/09/10.

Gajek, S., Jensen, M., Liao, L. and Schwenk, J. (2009), 'Analysis of signature wrapping attacks and countermeasures', *Proceedings of the IEEE International Conference on Web Services*, Los Angeles USA, pp. 575-582.

Gupta, A.N. and Santhi, T.P. (2016), 'Detection of XML signature wrapping attack using node counting', in Vijayakumar, V. and Neelanarayanan, V. (Eds), *Proceedings of*

the 3rd International Symposium on Big Data and Cloud Computing Challenges (ISBCC-16'). Smart Innovation, Systems and Technologies, Springer, Cham, Vol 49.

Jensen, M., Meyer, C., Somorovsky, J. and Schwenk, J. (2011), 'On the effectiveness of XML schema validation for countering xml signature wrapping attacks', *2011 1st International Workshop on Securing Services on the Cloud (IWSSC)*, IEEE, Los Angeles USA, July 9, pp. 7-13.

Kouchaksaraei, H.R. and Chefranov, A.G. (2013), 'Countering wrapping attack on XML signature in SOAP message for cloud computing', *International Journal of Computer Science and Information Security*, Vol.11 No. 9.

Kumar, J., Rajendran, B., Bindhumadhava, B.S. and Babu, N.S.C. (2017), 'XML wrapping attack mitigation using positional token', *2017 International Conference on Public Key Infrastructure and its Applications (PKIA)*, IEEE, Bangalore, November 14-15, pp. 36-42.

McIntosh, M. and Austel, P. (2005), 'XML signature element wrapping attacks and countermeasures', *Proceedings of the 2005 Workshop on Secure Web Services*, November, pp. 20-27.

Mell, P. and Grance, T. (2011), 'The NIST definition of cloud computing', *National Institute of Standards and Technology*, Vol. 53, No. 6, pp. 50.

Saudi, M.M., Zaizi, N.J.M., Sweese, K.J.A. and Bakar, A.A. (2019), 'Spatial Signature method (SSM) against XML signature wrapping attacks', *MATEC Web of Conferences 2019*, January 16, EDP Sciences, Vol. 255, # 02016.

Somorovsky, J., Heiderich, M., Jensen, M., Schwenk, J., Gruschka, N. and Lo Iacono, L. (2011), 'All your clouds are belong to us: Security analysis of cloud management interfaces', *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop*, New York USA, October 21, pp. 3-14.