字串限制式在行動應用程式安全檢測的應用

陳郁方
中央研究院資訊所
yfc@iis.sinica.edu.tw

摘要

根據 OWASP[1]統計，近年來注入式攻擊(injection attack)，在網頁程式和行動應用程式都是排名前十的攻擊方式。這類攻擊的原理，是攻擊者利用程式對於各種輸入來源沒有詳盡過濾的漏洞，藉機輸入程式原先設計外的可執行程式字串。執行這樣的字串，有可能造成各式各樣的損失。輕則遺失資料或系統損毀，重者甚至會泄露極重要的個人隱私資料。 字串限制式的解答技術，針對這樣的攻擊，可以有效率地找出潛在的漏洞並加以防治。我們相信是個很有潛力的新興方法。本文將對這個主題進行一個介紹性的探討。

關鍵詞：字串限制式，行動應用程式，HTML，注入式攻擊

壹、 前言

近年來注入式攻擊成為程式安全的一大隱憂。這樣的安全性問題可以說是無所不在。這問題的發生原因，是對於使用者輸入的字串，在使用(如組成 SQL 指令，或是送到 eval 函數)前沒有經過適當的過濾，導致意外執行了攻擊者所送入的程式碼。如果我們能有一個方法，對於程式所有可能的字串輸入，都做適當的檢測，過濾危險的輸入字串，這樣的問題就可以避免。然而，在行動裝置的環境下，可能的字串輸入方式實在層出不窮，從 QR code 掃描，無線網路基地台名稱，藍牙裝置名稱等，各種意想不到的攻擊方式一個個的被發掘。這樣的環境下，開發自動化的正規驗證方法，全面的檢查所有可能的字串輸入點，就成了一個重要的關鍵技術。而字串限制式(string constraints)的解答工具，就是字串操作程式自動化正規驗證方法的核心技術。

目前智慧型行動裝置主流的作業系統有三大陣營，為了開發能同時支援各種作業系統的應用程式，HTML5 是一個有效的程式語言。配合在各別作業系統上的 HTML5 執行平台，同樣的程式可以在很低的成本下支援所有的作業系統。然而，這也帶來了新的攻擊機會。傳統的跨站指令碼攻擊，現在也有可能在手機實現。甚至因為行動裝置的環境，還多了新的攻擊入口。字串限制式是檢測 HTML5 程式安全性很有潛力的方法。它有極高的正確性。透過這樣的技術，我們可以有效地找出程式的弱點並加以補強。在[5]中，針對於 HTML5 為基礎的行動應用程式可被攻擊的漏洞，有很詳盡的解

釋。為了本文的完整性，在這個章節中我們將簡述[5]提到的 HTML5 為基礎的行動應用程式安全問題。

根據 Evans Data 在 2012 的調查，75%的行動應用開發者，用 HTML5 作為他們的開發語言。但是這個技術是很容易遭受到駭客攻擊的。最主要的原因，是因為他們沒有清楚的區分程式碼和資料。這樣的特性有其方便性，但也是駭客最容易利用的攻擊入口。在傳統的網路應用程式上，跨站程式碼(Cross-Site Scripting, XSS)攻擊就是利用這個弱點。行動應用程式也有可能受到類似的方法攻擊。相對於網路應用程式，行動應用程式可以被攻擊的入口更多。根據[5]的研究結果，各式各樣的 bar-code(如 2D 的 QR-code)，簡訊，檔案系統中檔案的名稱，mp3 音樂的描述檔，WiFi access point 的名稱，通訊錄內容，藍牙裝置或事 NFC 裝置名稱等等，都是可以利用的攻擊點。一個可行的攻擊方式例子如下：攻擊者把有害的 JavaScript 程式碼包入 QR-code 中，當使用者用 HTML5 撰寫的行動應用讀取此 QR-code 並將內容印出。攻擊用的程式就有可能產生危險。熟悉網路應用開發的程式設計師也許會質疑，JavaScript 程式有這麼高的權限嗎。事實上，在行動應用，JavaScript 程式的權限往往高於在一般網頁程式。要了解細節，我們需要先談到行動應用程式的開發架構。

行動裝置的作業系統預設並沒有支援 JavaScript，為了執行 JavaScript 程式，以 Android 行動應用為例子，它必須用一個叫做 WebView 的元件來執行 JavaScript。在 iOS 中，相對應的元件叫做 UIWebView。在 Windows Phone 中對應的元件叫做 WebBrowser。這些元件通常是在沙箱(sand box)中執行。但是這樣對於行動應用而言侷限太多，因為行動應用常常需要用到外部的資源(如攝影鏡頭，簡訊，通訊錄等等)。為了應用這個需求，WebView 提供了呼叫沙箱外原生 Java 程式的界面。但是使用這些界面會造成不容易移植到其他平台(如從 Android 到 iOS)的問題。現在實務上的做法通常是透過一些中介軟體，例如 Adobe 的 PhoneGap，來呼叫沙箱外部的原生程式。PhoneGap 提供了 16 個預設的插件，讓程式設計師能夠使用行動裝置的各樣資源(如攝影鏡頭等)。在 PhoneGap 3.0 版以前(2013 年 10 月)，就算程式設計師用不到全部的 16 個插件，PhoneGap 還是把它們全部都開啓了。這導致了嚴重的後果，攻擊者甚至可以透過 JavaScript 程式，抓到行動裝置的 GPS 資料，並把它的位置上傳到指定的網站(詳見[5])。

要防止這類的攻擊，一個基本的原則，就是要阻止在沒有經過任何過濾的情況下，把從危險輸入界面(如前所提及的 bar-code，簡訊， 檔案名稱，mp3 描述檔等等)的資料，送到危險的輸出點(如 DOM API 的 document.write()，document.writeln()，DOM 屬性 innerHTML，outerHTML，jQuery API 的 html()，append()等等)。在[5]中，作者透過程式切片(program slicing)技術，取出行動應用原始碼中 JavaScript 相關的部分，再對該部分進行靜態分析(static analysis)。靜態分析是檢測這類危險行為是否存在程式中的一個工具。但是傳統的方法常常有精確性的問題，常常會有誤報的情況。如果能透過

字串限制式的方式來檢測，最大的好處，就是它具有 100%的精確性，完全不會有誤報的機會。

貳、 字串限制式和程式安全性的關係

我們先從一個簡單的例子看字串限制式和程式安全性問題的關係。下列程式(圖一)是一個典型注入式攻擊的例子。這是一個傳到客戶端的 HTML 程式，夾帶了 JavaScript 函數 checkTime 來檢查 input 的格式是否正確。當使用者在輸入時間的欄位鍵入一字串，checkTime 函數會被啟動來做基本的檢查。當檢查通過時，伺服器端的程式拿到輸入的時間，並且將內容 echo 出來，這樣的程式是安全的嗎。

```
<html>….<script>
    function checkTime(input) {
    var index = input.indexOf(":");
      var month = input.substr(index+1);
      return month.length<=2;
    }
 </script>  ...
    <form name="mainForm" action="/main" onsubmit="return checkTime(this);">
    Enter start time (hh:mm): <input type="text" name="start_time" size="64" />
    <input type="submit" value="Send" />
    </form>
... </html>
```

圖一：有弱點的程式片段

其實這樣的程式有很多被攻擊的機會。惡意的攻擊者可以寫一個夾帶 JavaScript 的時間字串，例如<script>…</script>:30。以這樣的方式可以夾帶任何的 JavaScript 程式碼到客戶端並執行。這樣的風險其實在行動應用程式也常常可以見到[5]。

就上面的例子，我們可以把程式是否會遭受到攻擊的問題，簡化到下列字串限制式是否有解答的問題： E=X:Y /\ |Y|<=2/\ X ∈ <script>.*</script> <註*>

我們用大寫字母表示字串變數，小寫字母和符號(:,/,<,>)表示字串常數，“<script>.*</script>”為正規表示式(regular expression)，代表了所有<script>開頭</script>結尾的字串，其中第一個部分式子 E=X:Y 代表的意思如下，使用者輸入的變數 E 等於兩個字串變數 X,Y 和常數:的連接 X:Y。第二部分指定長度變數|Y|的值小於等於 2，最

* 這個式子是簡化的版本，目的是讓大家看見字串限制式和程式碼之間的關係。第三章有較精確地介紹如何將程式碼轉為對應的字串限制式

後要求 X 的值在正規表示式<script>.*</script>裏面。如果這個字串限制式有解答，則其中 E 的值就是可行的攻擊字串。

然而字串限制式的解答是一個困難的工作。一個字串限制式總的來說可以包含下面三個種類限制式的任意布林組合：

(1)字串等式(word equation)：包含字串變數和常數的等式或不等式，這些變數的值可以是任意長度的字串，

例如：aX=Yb 或是 aX≠Yb，a 和 b 是字母常數，X 和 Y 是字串變數

(2)長度限制式(length constraint)：關於字串長度的限制式，

例如：|X|+|Y|<30，|X|和|Y|代表字串變數 X 和 Y 的長度

(3)範圍限制式(membership constraint)：字串變數範圍的限制式，範圍可以用正規表示式(regular expression)來表示

例如：X∈(a+b)*a，(a+b)*a 為一個正規表示式

這樣的組合已經足以表達大多數程式的行為。

這個問題最源頭的解答可以追朔到Makanine的演算法[6]。他提出第一個可以對任何由字串等式(word equation)布林組合形成的式子求解的演算法。然而，當考慮同時具有字串等式(word equation)和長度限制式(length constraint)布林組合的式子，這個問題究竟是不是可決定(decidable)的，目前的狀況還是未知的[2]。目前大部份人的做法，是在字串限制式中找一個夠大的子集，這個子集內的任何式子都是可解的。舉例來說，作者和瑞典合作的解答器Norn[1]所定義的子集，唯一的額外限制是字串等式中所有的字串變數只能在等號左邊出現一次。Norn解答器可以處理任何能被化簡到符合我們額外限制的式子。即使式子不符合這個限制，我們的解答器還是有機會能找到答案。我的競爭對手包括Z3-Str解答器[9][7]，他們方法最大的問題是他們不能處理範圍限制式。在驗證程式的時候，缺少範圍限制式，我們沒有辦法定義可能的攻擊字串。在之前的例子中，我們需要以下的式子X∈<script>.*</script>來限制變數X有問題的範圍。在[4]中所提出的演算法，有一個很大的限制。他們要求所有出現在範圍限制式的變數，只能出現在字串等式的左手邊。在這樣的限制下，有很多的程式行為都無法被描述。

在下面的章節，我們討論基於HTML5的行動應用程式和它的安全性問題。之後我們給一個對應，如何把檢查程式碼的安全性問題，化簡到字串限制式的解答問題。

參、 將 JavaScript 安全性問題化簡為字串限制式問題

要如何將 JavaScript 安全性的問題，化簡為字串限制式問題。原則上的做法會是將所有一行行的程式碼，轉成對應語意相同的字串限制式。由於本文是介紹性質，下面我們採用舉例的方式，列出一些常見的 JavaScript 字串操作方法[8]和它對應的字串限制式。對於這些轉換對應更系統化的描述，可以在[8]中找到。

例子一 charAt：S1:string=charAt(S:string,I:int)，拿到字串 S 中第 I 個字母，S 中第一個字母位置的索引是 0。如果 S 的長度小於等於 I，則回傳空字串。它可以對應到下列的字串限制式((I>=|S|\/ I<0/\S1= “”)\/(I<|S|/\0<I/\|S1|=1/\S=T1.S1.T2/\|T1|==I))，第一個部分的式子 I>=|S|\/ I<0/\S1= “” 說，如果 I>=|S|或是 I<0，則 S1 的值會是空字串。第二部分 I<|S|/\0<I/\|S1|=1/\S=T1.S1.T2/\|T1|==I 說，如果 I 的範圍正確，則 S 可以被分為三段 T1，S1，T2，其中 T1 的長度等於 I，S1 的長度等於 1。如此剛好 S1 就是 S 中索引為 I 的字母。整個式子要求第一部分或是第二部分其中之一要成立。

例子二 indexOf：I:int=indexOf(S:string, S1:string,I1:int)，回傳字串 S 中，從位置 I 開始算，子字串 S1 第一次發生的位置。如果在位置 I1 之後，S1 就沒有出現在 S 中，則回傳-1。這可以表達為下列的字串限制式：(|S|<I1/\I=-1) \/
(S=T1.T2/\I1=|T1|/\((T2=T3.S1.T4/\ T3 ∉.*S1.*/\I=|T3|+|T1|)\/ (T2 ∉.*S1.*/\I=-1)))
第一部分|S|<I1/\I=-1 描述當 I1 大於 S 的長度，則回傳值固定為-1。第二部分則是|S|>=I1 的情況，我們這時把 S 拆解為 T1 和 T2 兩部分，T1 的長度剛好為 I1。當 T2 有包含 S1 這個子字串，它可以被拆解為 T3，S1，T4，其中 S1 不可為 T3 的子字串(T3 ∉.*S1.*，.在正規表示式(regular expression)中為所有字元(all characters)的意思)。T3 的長度加上 T1 的長度，就是我們要的回傳值 I。當 T2 不包含 S1 這個子字串，則我們回傳-1。

例子三 concat：S:string=concat(S1:string,S2:string,...,Sk:string)，回傳 S1，S2，…，Sk 連接在一起的結果。這個情況可以輕易的用下列字串限制式描述 S = S1.S2.⋯.Sk。

例子四：substring：S:string=substring(S1:string,I1:int,I2:int)，回傳字串 S1 中，位置 I1 到 I2 的子字串(不包含 I2)。如果 I1 小於 I2，則互換這兩個數字。如果互換後 I2 大於 S1 長度，則取 I2 等於|S1|。我們用下列字串限制式表達 substring 方法：B=min(0,I1,I2)/\E=max(I1,I2,|S1|)/\S1=T1.S.T2/\|T1|=B/\|T2|=|S1|-E。我們用兩個整數變數，來記錄子字串開始和結束的位置。式子 S1=T1.S.T2 要求 S1 可以被分割為三部分。之後的式子為 T1 和 T2 對應的長度限制。

例子五 match：[S1, S2,...,Sk] : string list = match(S : string, re: Regex)，這是一個字串限制式不容易完美描述的例子。我們必須先假設對應到的字串有 k 個。而 k 是一個可以逐漸調高的常數。S1，S2，…，Sk 為 S 中對應到正規限制式 re 的子字串。我們用下列字串限制式來描述 S1，S2，…，Sk 和 S 的關係：S=T0.S1.P1/\P1=T1.S2.P2/\ … Pk-1=.Sk.Pk/\T0 ∉.*re.*/\S1 Ⅰre/\ T1 ∉.*re.*/\S2 Ⅰre/\.../\ Sk Ⅰre/\ Tk ∉.*re.*。在這式子中，S 可以被拆解成許多由 Ti 和 Si 所連結而成的子字串。Ti 為不包含任何符合 re 的子字串，Si 為符合 re 的子字串。

我們可以靠對變數的重新命名，來自動的處理連續的程式碼。舉例來說，我們可以將圖一中程式的 checkTime 方法，透過上述的轉換，變為字串限制式，為了方便閱讀，我們將 checkTime 方法內容也列在下面：

```
function checkTime(input) {
      var index = input.indexOf(":");
      var min = input.substr(index+1,input.length);
      return month.length<=2;
}
```

(INPUT=T1.T2/\I1=|T1|/\((T2=T3.S1.T4/\ T3 ∉ .*S1.*/\INDEX=|T3|+|T1|)\/ (T2 ∉ .*S1.*/\ INDEX =-1)))/\
B=min(0,INDEX+1, |INPUT|)/\ E=max(INDEX +1, |INPUT|,|INPUT|)/\INPUT=T5. M MONTH.T6/\|T5|=B/\|T6|=|INPUT|-E/\
RET=(|MONTH|<=2)

如果程式有迴圈，一個可行的方式，是透過克雷格內插法(Craig Interpolation)來找到可行的迴圈不變量。這個方法牽涉到一些程式驗證技術的細節，我們不在本文重述。有興趣的讀者，可以參考[1][3]。附帶一提，我們的工具 Norn，就可以替字串限制式找到克雷格內插法的值。

肆、 結論

透過字串限制式來解決 JavaScript 安全性的問題是一個新興的方法。此方法近年來在軟體安全，和自動化驗證頂尖的會議如 ACM CCS(Computer and Communications Security)，CAV(Computer-Aided Verification)，TACAS(Tools and Algorithms for the Construction and Analysis of Systems)受到廣泛的重視和討論。雖然到產品化，實用化還有距離。我們相信這是一個有潛力，十分值得進一步發展的技術。並鼓勵國內的學者共同加入此領域進行探索。

伍、 參考資料

[1] P. A. Abdulla, M. F. Atig, Y. F. Chen, L. Holík, Ahmed Rezine, Philipp Rümmer, Jari Stenman. String Constraints for Verification. CAV 2014: 150-166.

[2] J. R. Buchi and S. Senger. Definability in the existential theory of concate- nation and undecidable extensions of this theory. Z. Math. Logik Grundlagen Math., 34 (4), 1988.

[3] V. D'Silva, D. Kroening, G. Weissenbacher: A Survey of Automated Techniques for Formal Software Verification. IEEE Trans. on CAD of Integrated Circuits and Systems 27(7): 1165-1178 (2008).

[4] V. Ganesh, M. Minnes, A. Solar-Lezama, and M. Rinard. Word equations with length constraints: Whats decidable? In Hardware and Software: Verification and Testing, volume 7857 of Lecture Notes in Computer Science, pages 209– 226. Springer Berlin Heidelberg, 2013.

[5] X. Jin, X. Hu, K. Ying, W. Du, H. Yin, and G. N. Peri. 2014. Code Injection Attacks on HTML5-based Mobile Apps: Characterization, Detection and Mitigation. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS '14).

[6] G.S. Makanin. The problem of solvability of equations in a free semigroup. Mathematics of the USSR-Sbornik, 32(2):129–198, 1977.

[7] L. D. Moura and N. Bjorner. Z3: An efficient SMT solver. In Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08/ETAPS'08, pages 337–340, 2008.

[8] P. Saxena, D. Akhawe, S. Hanna, F. Mao, S. McCamant, D. Song:A Symbolic Execution Framework for JavaScript. IEEE Symposium on Security and Privacy 2010: 513-528.

[9] Y. Zheng, X. Zhang, and V. Ganesh. Z3-str: A Z3-based string solver for web application analysis. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2013, pages 114–124, New York, NY, USA, 2013. ACM.

[10] http://www.eweek.com/c/a/Application-Development/75-of-Developers-Using-HTML5-Survey-508096

[11] https://www.owasp.org/index.php/OWASP_Mobile_Security_Project#tab=Top_10_Mobile_Risks.