正規表示式阻斷服務（ReDos）攻擊

資料來源：HP 提供 | 翻譯整理：叡揚資訊 資訊安全事業處

正規表示式阻斷服務（ReDos）其實不是什麼新的技術，但並未被重視。正規表示式使用起來常常會犯些小錯誤，而這對攻擊是非常有效的，有時甚至難以解決。

容易發生的錯誤

簡單的說，任何正規表示式的 pattern 會有重複的符號，如「+」、「*」，表示可重複的部份(但這不是唯一有弱點的 pattern，可參考以下其他有弱點的 pattern)，且這類的問題是滿常見的，連 OWASP 都有收錄此問題，並建議如果達到驗證 pattern，可參考：

http://en.wikipedia.org/wiki/ReDoS#Vulnerable_regexes_in_online_repositories

常見弱點 pattern：

- (a+)+
- ([a-zA-Z]+)*
- (a|aa)+
- (a|a?)+
- (.*a){x} for x > 10

攻擊範例

以下為幾個攻擊測試，pattern 為「^(a+)+$」，同時測試 Java 與.Net，結果如下：

Input	Java Pattern.matches("^(a+)+$", input);	.NET new Regex("^(a+)+$").IsMatch(input)
aaaaaaaaaaaaaaaaaaaa!	79ms	299ms

GSS Galaxy Software Services 叡揚資訊
台北總公司：10461 台北市中山區德惠街 9 號 5 樓 TEL：(02)2586-7890 FAX：(02)2586-8787
高雄辦公室：80453 高雄市明華路 315 號 13 樓之 3 TEL：(07)586-6195
海外據點：上海、北京 營業範圍：兩岸三地及日本 www.gss.com.tw

aaaaaaaaaaaaaaaaaaaa!	144ms	542ms
aaaaaaaaaaaaaaaaaaaaa!	259ms	1113ms
aaaaaaaaaaaaaaaaaaaaaa!	434ms	2139ms
aaaaaaaaaaaaaaaaaaaaaaa!	867ms	4509ms
aaaaaaaaaaaaaaaaaaaaaaaa!	1823ms	8672ms

正如您看到的，長度 25 個字元時，會讓執行序忙碌長達 1 秒，而每多加一個字元所花費的時間是以倍數成長。且不要以為攻擊字串使用一連串的「a」是很容易過濾，這一切都需仰賴驗證 pattern 的設計，是否能辨別看起來像個正常的字串，但其實是個攻擊行為。所以不易透過簡單的驗證來阻止此類型的攻擊。

不易修復

OWASP 提到的弱點 pattern，是以 Java 為例檢測類別名稱「^(([a-z])+.)+[A-Z]([a-z])+$」。這不是一個非常複雜的 pattern，但該如果調整?注意實際上真的有可能類別名稱是長於 25 個字元，其實不是那麼容易解決。並需妥協於 false positives(誤報) 與 false negatives(誤判)。

其他解決辦法

有些版本的 Perl PCRE 函式庫有個「match」函式，當執行 pcre_exec()與 pattern 匹配時呼叫遞迴。通過控制每次執行操作時可被呼叫的最大次數，限制所使用的資源。而限制的數量被設定於 pcreapi 文件中。預設值為 10 萬，可直接修改數值或也可透過命令修改如「--with-match-limit=500000」。

GSS 叡揚資訊 Galaxy Software Services
台北總公司：10461 台北市中山區德惠街 9 號 5 樓 TEL：(02)2586-7890 FAX：(02)2586-8787
高雄辦公室：80453 高雄市明華路 315 號 13 樓之 3 TEL：(07)586-6195
海外據點：上海、北京 營業範圍：兩岸三地及日本 www.gss.com.tw