ReDoS 脆弱な正規表現の修正

富家　功一朗

早稲田大学基幹理工学部情報理工学科

January 27, 2023

目次

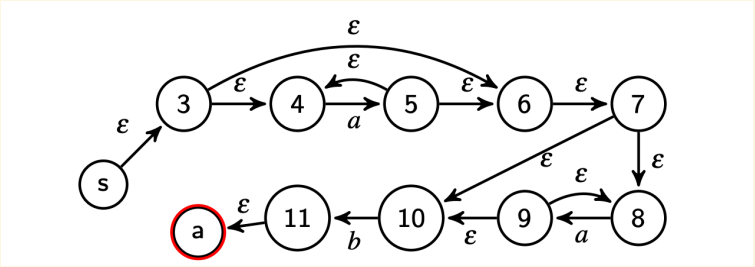
- ReDoS とは何か？
- regular expression denial of service の略．
- 正規表現のパターンマッチの脆弱性を利用した DoS 攻撃 (ウェブサイトやサーバーに負荷をかけ，その可用性を侵害する行為) のこと．

ReDoS 脆弱性を孕んだ正規表現を用いた結果，以下のような事例が発生している．

- 2016 年，Stack Overflow のサーバーが，空白を 2 万個含む投稿によりダウンした [1]．
- 2019 年，Cloudflare が提供する CDN が，ファイヤーウォールに追加した脆弱性な正規表現によりダウンした [2]．

では，正規表現のパターンマッチの脆弱性とは一体何なのだろうか？

- そもそも正規表現のパターンマッチ処理はその正規表現に対応する NFA をバックトラック探索することにより行われている．



- 上記の図は正規表現 a^*a^*b に対応する NFA である ([3] より引用)．
- aaa という文字列は a^*a^*b に受理されないが，このときどのようなパターンマッチ処理が発生するのかを見る．

- まず，初期状態 s から遷移できるところまで進む．今回の例では初期状態 s から a という文字を 2 つ消費した後の状態 10 までは遷移できる．以下はそのようなパス．

$$(s, \varepsilon, 3), (3, \varepsilon, 4), (4, a, 5), (5, \varepsilon, 6), (6, \varepsilon, 7), (7, \varepsilon, 8), (8, a, 9), (9, \varepsilon, 10) \quad (1)$$

- しかし，状態 10 から先へは文字 b がなければ進めない．
- このような状況に陥った際，1 手前に戻る，すなわちバックトラックが行われる．
- 具体的に言えば，(1) の探索パスで言えば状態 10 から進めないことが分かったので状態 9 に戻ってやり直すということである．状態 9 に戻って，残りの文字 a を読み取った上で状態 a にたどりつけるようなパスを探すが存在しないので，またバックトラックする．

このように文字列のパターンマッチングではバックトラックを繰り返し行うことが多々ある．そして，バックトラックの回数が文字列のパターンマッチングの実行時間に大きな影響を与える．

ここからは脆弱な正規表現を修正する方法について見ていく．まず，本研究で提案する手法では REMEDY，RegEq という 2 つのプログラムを利用するのでそれらの説明を行う．

ベースとなるプログラムの説明

まず，RegEq[4] について説明を行う．

2 つの正規言語を入力としたとき，それらが意味的に等価であるかを判定する．意味的に等価というのは，受理する文字列の集合が等しいということ．等価でなかった場合，片方の正規言語には受理されるが，もう片方の言語には受理されない文字列が反例として返される．

RegEq では 2 つの正規言語を DFA に変換し，差を取り，それが空であるかどうかを判定することにより，等価性判定が行われている．

ベースとなるプログラムの説明

次に，REMEDY[5] について説明を行う．

REMEDY は正規表現と positive example(正例) と negative example(負例) を入力として与えると，以下の条件を満たすような正規表現を出力として返す．

1. 入力として与えられた positive example，negative example が正しく分類されている (positive example は受理され，negative example は受理されない)
2. real-world strong 1-unambuiguity(RWS1U)
3. 入力として与えられた正規表現に構文的に近い

- RWS1U を満たせば正規表現が非脆弱であることが保証される．
- あくまで修正後の正規表現が与えられた例を判別する能力を持つだけで，修正前の正規表現と後の正規表現が意味的に等しいとは限らない．

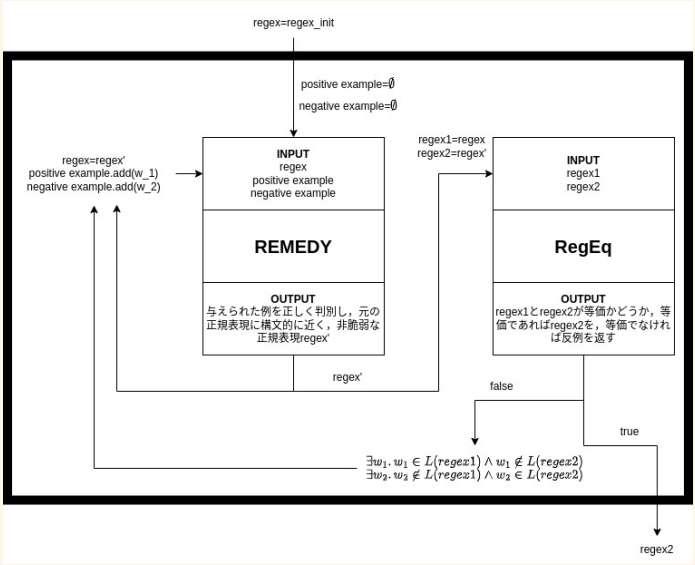
提案手法

上記の 2 つのプログラムを用いて以下のようなアルゴリズムで正規表現を意味的な等しさを保ったまま非脆弱化する．

1. (ReDoS 脆弱だと疑われる) 正規表現を REMEDY への入力とする．はじめは positive example，negative example ともに入力として渡さない (つまり positive example の集合，negative example 集合がともに空である)．
2. REMEDY から出力された正規表現と元の正規表現を RegEq への入力とし，
   - 等価であるなら修正後の正規表現を返してこのアルゴリズムは終了．
   - 等価でないなら元の正規表現に受理され修正後の正規表現に受理されない文字列を正例に追加し，元の正規表現に受理されず修正後の正規表現に受理される文字列を負例に追加し，それらを修正後の正規表現とともに REMEDY に渡す．
3. 2. を繰り返す．

なお，このアルゴリズムは停止しない場合がある．なぜなら，ある正規表現に対してそれを表しかつ S1U を満たすような正規表現がない正規言語が存在するからである．

上記のアルゴリズムを直感的に表した図を以下に示す．



実験

脆弱な正規表現 $c(ab)^*a(ba)^*$ と $.^*a^*=$ に本アルゴリズムを適用してみる．その結果は以下の通り．

Table 1: 脆弱な正規表現 $c(ab)^*a(ba)^*$ の修正結果

修正前の正規表現	修正後の正規表現	修正にかかった時間 [s]
$c(ab)^*a(ba)^*$	$ca(ba)^*$	0.533

Table 2: 脆弱な正規表現 $.^*a^*=$ の修正結果

修正前の正規表現	修正後の正規表現	修正にかかった時間 [s]
$.^*a^*=$	$[\hat{}=]^*=([\hat{}=]^*=)^*$	55.000

- 今回は比較的短時間で終わる例のみ取り上げた．1 時間かけても終わらない例もあった．
- REMEDY は文字列例を元に正規表現を構成する PBE(programming by example) という手法を取っているため，適切な例を持ってこれるか否かが本プログラムの実行速度に影響を及ぼす．
- だが，どのような例を持ってくれば早く目的の正規表現を得られるのかはよく分かっていないので今後の課題としたい．

Reference I

[1] Stack Exchange Network Status — Outage Postmortem - July 20, 2016.
https://stackstatus.tumblr.com/post/147710624694/outage-postmortem-july-20-2016.
Accessed: 2023-01-21.

[2] Details of the cloudflare outage on july 2, 2019.
https://blog.cloudflare.com/details-of-the-cloudflare-outage-on-july-2-2019/.
Accessed: 2023-01-21.

[3] Cristian-Alexandru Staicu and Michael Pradel.
Freezing the web: A study of redos vulnerabilities in javascript-based web servers.
In *Proceedings of the 27th USENIX Conference on Security Symposium*, 2018.

[4] Regeq.
https://bakkot.github.io/dfa-lib/regeq.html.
Accessed: 2023-01-21.

[5] Nariyoshi Chida and Tachio Terauchi.
Repairing dos vulnerability of real-world regexes.
2020.