Article
Danny Wijnschenk · Nov 7, 2017 5m read

Advent of Code 2016 Day7: Internet Protocol Version 7

This is a series of programming challenges for beginners and experienced Caché programmers.

For an introduction : goto to article https://community.intersystems.com/post/advent-code-2016-day1-no-time-ta...

Today's challenge on http://adventofcode.com/2016/day/7 is about checking for valid IPv7 addresses with TLS support. (No, it has nothing to do with real ip addresses which are at most ipV6 or real TLS, but just a way to keep you busy coding & hacking!)

The imaginary ipV7 addresses support TLS if they contain an ABBA sequence outside square brackets and no ABBA sequence inside brackets. (ABBA : any 4 character sequence where the first pair are 2 different characters followed by the reverse of that pair).

For example :

```
abba[mnop]qrst : supports tls
abcd[bddb]xyyx : no support for tls
aaaa[qwer]tyui : no support for tls
ioxxoj[asdfgh]zxcvbn : supports tls
```

Make a routine which can detect support for TLS in addresses found here:
http://adventofcode.com/2016/day/7/input

My solution just takes every part of an address and checks for ABBA. Since there can be multiple bracket parts, I loop over each character in the address and stop at each [or] to test the part. (I could replace the bracket by a single delimiter with one $Replace, and than loop over it with $Piece ..., or do the same with $ListBuild which would be a little bit faster)

Here is my try to solve this puzzle :

```
Class AOC2016.Day7 Extends AOC2016.Utils
{

ClassMethod Part1(file As %String = "day7.txt")
{
  #Dim input, ipAddress, part as %String
  #Dim iInput, iIp as %integer
  #Dim tlsCount as %Integer = 0 //counter of TLS compatible addresses
  #Dim okTls as %Boolean
  Try {
    Do ..Input(file, .input)
    For iInput=1:1:input {
      Set ipAddress=input(iInput)
      Set part=""
      Set okTls = ""
      For iIp=1:1:$Length(ipAddress) {
```

```
        If $E(ipAddress,iIp)="[" {
          If part'="" Do ..CheckTLS(part,0,.okTls) ;0=part outside bracket
          Set part=""
        } elseIf $E(ipAddress,iIp)="]" {
          If part'="" Do ..CheckTLS(part,1,.okTls) ;1=part within bracket
          Set part=""
        } else {
          set part=part_$E(ipAddress,iIp)
        }
        If okTls = 0 Quit  //cannot get true anymore
      }
      If part'="", okTls'=0 Do ..CheckTLS(part,0,.okTls)
;check the remainder which is always outside bracket
      If okTls = 1 set tlsCount=tlsCount+1
    }
  } Catch {
    Write "Error : ",$ZError,!
  }
  Write "TLS : ",tlsCount,!
}

ClassMethod CheckTLS(part As %String, inBracket As %Boolean, ByRef ok As %Char)
{
  #dim fourSequence as %String = ""
  #dim sequence as %Integer = 0
  #dim iPart as %Integer
  //Check if part of address has any ABBA sequences
  For iPart=4:1:$Length(part) {
    Set fourSequence=$E(part,iPart-3,iPart) ;last 4 characters
    //check for ABBA
    If ($E(fourSequence,1,2)=$Reverse($E(fourSequence,3,4))) & ($E(
fourSequence,1)'=$E(fourSequence,2)) {
      Set sequence=1
      Quit
    }
  }
  If sequence & inBracket {
    Set ok = 0 //any ABBA within brackets makes the address not support TLS
  } elseIf sequence & 'inBracket {
    If ok'=0 Set ok = 1
//a ABBA outside brackets makes the address support TLS if no ABBA's were found withi
n brackets (ok is initialized as "")
  }
}

}
```

The second part of the puzzle is about detecting addresses with SSL support (again, any resemblance with real acronyms is pure by concidence).
An address supports SSL if you find ABA sequences outside of square brackets, and a corresponding BAB sequence inside square brackets.

For example :

```
aba[bab]xyz supports SSL
xyx[xyx]xyx does not support SSL
aaa[kek]eke supports SSL
```

Bert is doing a slightly different approach by looping only once through the address characters.

Here is his code for the second part :

```
Start2() PUBLIC {
  #Dim objFileStream As %Stream.FileCharacter
  s objFileStream = ##Class(%Stream.FileCharacter).%New()
  s sc=objFileStream.LinkToFile("J:\Winfo\AOC\day7.txt")
;"C:\Users\15274\workspace\adventofcode\2016\input\day07\input.txt")
  s count=0
  while 'objFileStream.AtEnd {
    k inHyper
    k outHyper
    s supportsSSL=0
    s inHypernet=0
    s line=objFileStream.ReadLine()
    f i=1:1:($LENGTH(line)-2) {
      s possible=$EXTRACT(line,i,i+2)
      if $EXTRACT(possible)="[" {
        s inHypernet = 1
      }
      if $EXTRACT(possible)="]" {
        s inHypernet = 0
      }
      if (($EXTRACT(possible)=$EXTRACT(possible,3)) && ($EXTRACT(possible,1)'=
$EXTRACT(possible,2))) {
        if inHypernet {
          if $DATA(outHyper($EXTRACT(possible,2)_$EXTRACT(possible,1)_
$EXTRACT(possible,2))) {
            s supportsSSL=1
            w !,possible
            w !,line
          }
          s inHyper(possible)=""
        } else {
          if $DATA(inHyper($EXTRACT(possible,2)_$EXTRACT(possible,1)_$EXTRACT(
possible,2))) {
            s supportsSSL=1
            w !,possible
            w !,line
          }
          s outHyper(possible)=""
        }
      }
    }
    if supportsSSL {
      s count = count +1
    }
  }
  w !,count
}
```

We are now doing this challenge for one week, and i get the feeling the more difficult challenges are still to come.

Look here for all our solutions so far : https://github.com/DannyWijnschenk/AdventOfCode2016 and https://bitbucket.org/bertsarens/advent2016.

Here is the list of all Advent of Code 2016 articles :

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

#Caché #Code Snippet #Contest #ObjectScript

Source URL: https://community.intersystems.com/post/advent-code-2016-day7-internet-protocol-version-7