DeMarcus Campbell

April 10, 2019

Homework 6

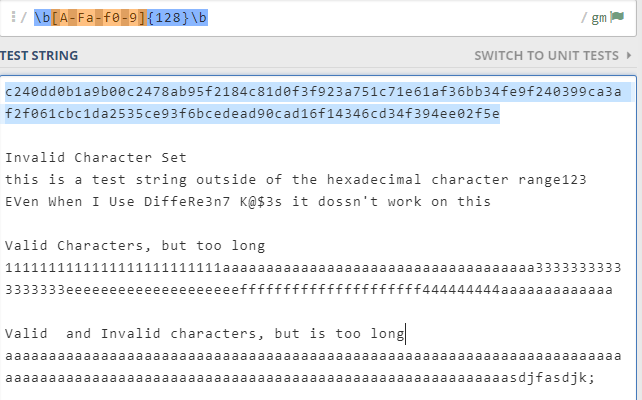
COMP 5350

**SHA-512**

This hashing algorithm outputs a 128 character long hexadecimal string. To create a regular expression that matches this specification, we must search for the valid hexadecimal characters in groups of 128 characters. That expression should look like this:

\b[A-Fa-f0-9]{128}\b

The above regular expression searches for the valid hexadecimal characters in 128-character chunks. The following picture shows a string encoded using the SHA-512 hash, and tests the regular expression against it and some other non-valid strings.

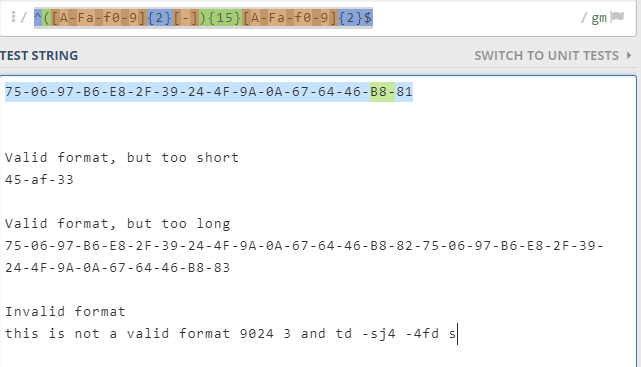


**LANMAN Hash**

The LANMAN hash is a Legacy Microsoft hashing algorithm used by Microsoft LAN Manager and Windows versions before NT. They used this hashing algorithm to store their passwords. This hashing can be recognized by a hyphenated string of 16 hexadecimal numbers. A regular expression that can match this style of hash is:

^([A-Fa-f0-9]{2}[-]){15}[A-Fa-f0-9]{2}$

This regular expression checks for 15 hexadecimal bytes followed by hyphen characters, and then checks that the last portion of the string is a hexadecimal byte. This regular expression also ensures that it only returns true for strings that start and end in our desired format. This ensures that it doesn’t return true for strings in the proper format but are too long.

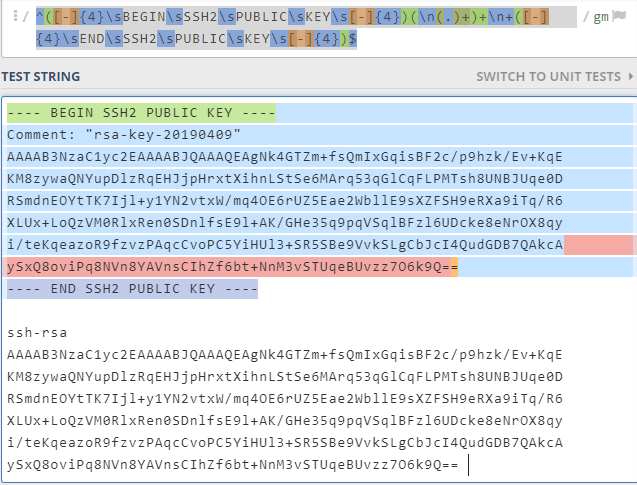


**SSH Public Key**

SSH public keys are used to allow a user to login to a server without a password via SSH. This password-less login only works if the user has access to the private key file that corresponds to the public key on the server. SSH public keys tend to start and end with a series of hyphens then a string stating that it is the beginning or end of the key. An important thing to note is that depending on how the key was generated, it will not always have the beginning and ending flags, but rather a string stating the encryption type. A regular expression that should work for a SSH public key in the “flag” format is:

^([-]{4}\sBEGIN\sSSH2\sPUBLIC\sKEY\s[-]{4})(\n(.)+)+\n+([-]{4}\sEND\sSSH2\sPUBLIC\sKEY\s[-]{4})$

This regular expression searches for the flag “---- BEGIN SSH2 PUBLIC KEY ----” then allows for any amount of characters, including newline, to be in between until it finds the closing flag “---- END SSH2 PUBLIC KEY ----.” The following image shows how this will work for an SSH public key in the “flag” format, but not for the OpenSSH format.



NOTE: It is possible to create regular expression that will both formats, but it is long

**SSH Private Key**

SSH private keys, in conjunction with SSH public keys, allow users to login to a system without a password given that the user supplies the private key to one of the system user’s public key. There are two main formats of private keys as with public keys. They are OpenSSH and PuTTY. The Putty format can be characterized as having both the public and private key in a single file separated by descriptor strings. A regular expression that will work for the PuTTY format is:

^(PuTTY-User-Key-File-2:\s).+\n(.+\n){2}(Public-Lines:\s)\d+\n((.+)\n)+(Private-Lines:\s)\d+\n((.+)\n)+(Private-MAC:\s)[A-Fa-f0-9]{40}$

This regular expression specifically is designed for the PuTTY format of private keys and should not work for any other type of file, as it searches for specific strings with certain characters following, whether that be whitespace, a number, or a MAC address depends on the line. The picture below illustrates how this regular expression returns true for the entirety of a PuTTY format private key.

