Sécurité des Réseaux - TP1

Cryptographie

Monroe Samuel

30 décembre 2015

1 Implémentation du Code de César

La partie la plus difficile dans une implémentation brute-force, à mon avis, est le fait de savoir quel résultat est pertinant.
Dans ce cas-ci c'est assez simple, mais pour d'autres codages les possibilités peuvent être beaucoup plus nombreuses et selon ma méthode, prendraient beaucoup plus de temps à être analysées.

Mon Crack aurait pu être amélioré en implémentant une analyse des résultat, via comparaison avec un dictionnaire de mots courants, et en sélectionnant le résultat ayant le plus de correspondance avec ce dictionnaire.

```
const firstLowerCase = 97;
const firstUpperCase = 65;
const lastLowerCase = 122;
const lastUpperCase = 90;

/**
  * @param message : String to encrypt or decrypt
  * @param offset : Int value used for the Cesar code
  * @param code : Boolean values, true encrypt the message, false decrypt it
  * @return String result of the operation
*/
function cesar(message, offset, code){
  var msgLen = message.length;
  var output = "";
  for(var i = 0; i < msgLen; i++){
    var newAscii = message[i].charCodeAt(0);
    if(is_alphabetic(newAscii)){
      for(j = offset; j > 0; j--){
        if(code){
          if(newAscii == lastLowerCase) newAscii = firstLowerCase;
          if(newAscii == lastUpperCase) newAscii = firstUpperCase;
          newAscii++;
        } else {
          if(newAscii == firstLowerCase) newAscii = lastLowerCase;
          if(newAscii == firstUpperCase) newAscii = lastUpperCase;
          newAscii--;
        }
      }
    }
    output += String.fromCharCode(newAscii);
  }
  console.log(output);
}

/**
  * @param message : String to crack by brute-force method
  * @return Strings, displaying all possibilities
*/
function cesar_crack(message){
  var msgLen = message.length;
  for(var abc = 1; abc <= 26; abc++){
    var output = "";
    for(var i = 0; i < msgLen; i++){
```

```
        var newAscii = message[i].charCodeAt(0);
        if(is_alphabetic(newAscii)){
          for(j = abc; j > 0; j--){
            newAscii++;
            if(newAscii == lastLowerCase + 1) newAscii = firstLowerCase;
            if(newAscii == lastUpperCase + 1) newAscii = firstUpperCase;
          }
        }
        output += String.fromCharCode(newAscii);
      }
      console.log("Tryout #" + abc + " : " + output);
    }
}

/**
  * @param ascii : Character
  * @return True if Char is alphabetic, false if a special Char
*/
function is_alphabetic(ascii){
  if((ascii >= firstLowerCase && ascii <= lastLowerCase) || (ascii >= firstUpperCase &&
  else return false;
}
```

2 John the Ripper

2.1 Shadow, Unshadow

Le fichier de mots de passes que nous récupèrons sous Linux dans /etc/shadow est lié à l'utilitaire Shadow.

De base, les mots de passes hachés du système sont stockés à l'emplacement **/etc/passwd**, de fait ces informations sont accessibles à quiconque aurait réussi à s'introduire sur la machine, et lui permettrait de tenter de les cracker.

Shadow permet de transposer ce fichier sous le nom **shadow** dans un emplacement accessibles uniquement en disposant des droits de Root, ce qui réduit considérablement les risques d'accès à ce fichier sensible.
Unshadow permet de sortir de l'ombre ce fichier pour une quelconque raison, ici expérimenter sans entrave le programme John.

2.2 Type de chiffrement

Sur notre système CentOS 6, un **cat** du fichier shadow nous renseigne sur le type de mécanisme cryptographique utilisé pour hasher les mots de passes.
On retrouve en début de ligne des hash un signe $ associé à un numéro, ici le 6, qui signifie l'utilisation de l'algorithme de hashage **SHA-512**.

2.3 Mesures

Les mots de passes dérivés du username sont trouvés presque **instantanément** en utilisant JTR de façon classique (trois modes successifs) ou via en spécifiant le mode simple.

La recherche via dictionnaire peut durer jusqu'à 45 minutes en utilisant le dictionnaire fourni sur CentOS.

La durée de la recherche incrémentale est directement liée à la complexité du mot de passe, plus celui-ci est long et varié en termes de caractères (alphanumériques, signes spéciaux, espaces), plus il est sûr.

On en concluera que les mots de passes dérivés du username sont les plus faibles, ceux tirés des mots courants sont aussi inutiles, les meilleurs mots de passes sont les plus complexes et les plus longs.