Praktikumsaufgabe 2
Kryptographische Verfahren und Anwendungen II
Sommersemester 2015

Da das in Aufgabe 3 verwendete Programm „John the Ripper“ vom Windows Defender als gefährlich eingestuft wird, sollten Windows Benutzer diese Aufgabe in einer Virtuellen Maschine mit dem Betriebssystem Kali bearbeiten. Eine Anleitung wie Ihr eine Virtuelle Maschine mit Kali aufsetzt oder wie Ihr Kali als Live-System bootet findet Ihr im Moodle. ***Falls Ihr keinen geeigneten Rechner zur Verfügung habt, könnt Ihr gerne einen unserer Laborrechner verwenden – sprecht uns dafür rechtzeitig an.***
Generell ist es empfehlenswert Aufgabe 3 dieser Übung in einer Virtuellen Maschine zu bearbeiten, da man durch Beschränkung der Ressourcen einer VM nicht Gefahr läuft sein System zu überlasten.

Aufgabe 1: *KDFs und Password Based KDFs (Theorie)*

In der folgenden Abbildung ist eine vereinfachte Version einer standardisierten KDF nach NIST SP800-108 (siehe Link im Moodle) - Kapitel 5.1 - dargestellt.

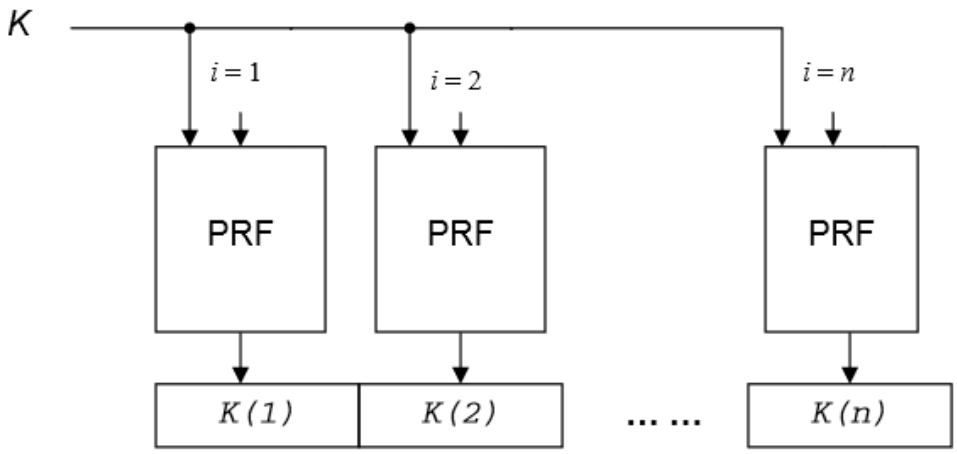

Abbildung 1

a) Beschreiben Sie mit Hilfe von Abbildung 1 und NIST SP800-108 die Funktionsweise dieser KDF.
b) Unter Betrachtung von Kapitel 4 des Standards: Welche Primitive die Sie in *KVA I* kennengelernt haben eignen sich als Kandidaten für die PRF (zusätzlich zu den dort genannten)?
c) An welche Ihnen bekannten Konstruktionen erinnern Sie die im Standard beschriebenen KDFs?
d) Warum eignet sich diese Konstruktion nicht mehr zur Schlüsselableitung, wenn der Schlüssel **K** aus Abbildung 1 durch ein Passwort ersetzt wird?

Aufgabe 2: *Passwort Speicherung (Praxis)*
Sie können eine Programmiersprache Ihrer Wahl zum Lösen der Aufgabe verwenden und Kommandozeilentools wie openssl zur Hilfe nehmen (in Kali bereits vorhanden).

a) Die Datei *pwListClear.txt* beinhaltet eine Liste von Benutzernamen und Passwörtern im Klartext getrennt mit einem $ Symbol. Schreiben Sie ein Programm, welches eine neue Liste mit dem Namen *pwListSha1.txt* erzeugt, wobei alle Einträge die Form *username:SHA1(password)* haben sollen (der Hash sollte dabei in Hexadezimal gespeichert werden).
b) Fällt Ihnen beim Betrachten der Liste in *pwListSha1.txt* etwas auf? Welche Informationen würde ein Angreifer erhalten wenn man Passwörter auf diese Art speichert?
c) Um das Problem aus 2b) zu lösen ändern Sie ihr Programm aus Aufgabe 2a) wie folgt ab: Für jeden Eintrag in der Liste wird ein zufälliger Wert *salt* (128Bit) bestimmt. In der neuen Datei *pwListSha1Salt.txt* soll jeder Eintrag folgende Form haben (Salt und *SHA1(password||salt)* in Hex):

   username :salt : *SHA1(password||salt)* (|| bedeutet Konkatenation)

d) Angenommen ein Server speichert die Passwörter seiner Benutzer so wie in Aufgabe 2c). D.h. ein Benutzer registriert sich einmalig mit seinem Passwort und der Server generiert einen Salt und speichert den „gesalzenen" Hashwert ab. Wie kann sich ein Benutzer zu einem späteren Zeitpunkt am Server authentifizieren ohne sein Passwort zu übertragen?

Aufgabe 3: *Wörterbuch-Angriffe (Praxis)*
Die Datei *passwd* beinhaltet 7 Passwörter, dazugehörige Benutzernamen und weitere Benutzerinformationen. Dabei sind die Passwörter mit der Linux Funktion crypt(C) gehasht worden, welche einen Salt und SHA512 verwendet.

a) Verwenden Sie die Online Dokumentation von John the Ripper und beschreiben Sie den „Single-Mode" von John in Ihren eigenen Worten.
b) Probieren Sie John im Single Mode mit der Datei *passwd* aus (John kann jederzeit mit Strg+C unterbrochen werden):

   john –single passwd

   Wie ist das Passwort des Benutzers *bananajoe*? Das Ergebnis können Sie sich mit *john –show passwd* noch einmal anzeigen lassen.
c) Wie konnte John das Passwort des Benutzers *bananajoe* brechen? Welche Informationen haben dabei geholfen?

Für die restlichen Passwörter benötigen wir einen mächtigeren Angriff: den Wörterbuchangriff. Dazu müssen Sie zunächst ein Wörterbuch erstellen. Die Grundlage bildet

die Datei *passwordList.lst* welche lediglich 10 Passwörter enthält. Diese Liste werden Sie mit Hilfe von Regeln erweitern.

d) Die John config Datei beinhaltet alle Regeln die John zur Erzeugung von Wörterbüchern verwendet. Wechseln Sie in das Verzeichnis */etc/john.* Dort machen Sie zunächst eine Sicherung der Datei *john.conf (⇒ john.conf.backup).* Jetzt kopieren Sie die Datei *john.conf* aus dem Übungsmaterial in den Ordner */etc/john*.

e) Wechseln Sie wieder in das Verzeichnis in dem Ihr Übungsmaterial liegt. Um zu überprüfen, ob Sie im vorherigen Schritt alles richtig gemacht haben erzeugen Sie eine neue Passwortliste mit: *john –wordlist:passwordList.lst –stdout –rules > test.lst*
Enthält die Datei *test.lst* die Passwörter auch mit großem Anfangsbuchstaben ist alles in Ordnung.

f) Nun können Sie mit der Bearbeitung der Regeln beginnen. Dazu müssen Sie die Datei *john.conf* abändern. Die Kommentare innerhalb der Datei sollten Ihnen dabei helfen, die Syntax der Regeln finden Sie unter dem Link im *Moodle*.

john –wordlist:passwordList.lst –stdout –rules > passwdList.lst

Die Regeln für unser Wörterbuch finden Sie in Tabelle 1. Überlegen Sie sich zunächst eine geschickte Reihenfolge für diese Regeln. Beachten Sie dabei, dass Sie sich auch ein Übergangswörterbuch erzeugen können, indem Sie nur einige Regeln anwenden. Danach können Sie weitere Regeln auf dieses Wörterbuch anwenden.

Grundlegende Regeln:
Alle Passwörter vollständig groß schreiben
Den Anfangsbuchstaben aller Passwörter groß schreiben
Ersetzungsregeln:
Ersetze a durch @
Ersetze e durch 3
Präfix Regeln:
Alle Zahlen von 0-9 als Präfix
Die Zeichen ?! als Präfix
Suffix Regeln:
Alle Zahlen von 0-9 hinter dem Wort anhängen
Die Zeichen ?! hinter dem Wort anhängen
Kombinierte Regeln
Alle Zahlen von 0-9 als Präfix und hinter dem Wort anhängen

Tabelle 1

g) Versuchen Sie mit Hilfe Ihrer Passwortliste die restlichen Passwörter zu knacken. Dafür führen Sie folgenden Befehl aus: *john –wordlist:passwdList.lst passwd*

Falls Sie nicht alle Passwörter brechen können sollten Sie das Wörterbuch noch einmal überarbeiten. [Für die erfolgreiche Teilnahme ist es nicht notwendig alle Passwörter zu brechen]

Aufgabe 4: *Abgabe*

Senden Sie Ihre Lösungen von Aufgabe 1, Ihr Programm/Script und die Dateien *pwListSha1.txt pwListSha1Salt.txt* aus Aufgabe 2, sowie aus Aufgabe 3 Ihre john.conf Datei und Ihr finales Wörterbuch mit einem Screenshot von Johns Crackingergebnis (john –show passwd) in einer mit PGP signierten und verschlüsselten E-Mail an:

- thomas.koller@uni-siegen.de (KeyID: 3695ED66, Fingerabdruck: 4E719EF64015F40013416DAC739EB27E3695ED66)
- robin.fay@uni-siegen.de (KeyID: 5900F665, Fingerabdruck: 0EAE1B2C9DBAAAF28DE75D5E3FF08B2C5900F665)