Bezpieczeństwo Lista 2

Bartosz Michalak

23 March 2024

1 Zadanie 1

1. We create password and push it to file like so:

```
$ echo -n "pass_phrase" | md5sum | awk '{print $1}' > pass_phrase.txt
```

Where

```
echo -n "pass_phrase"
```

outputs pass_phrase without new line at the end (-n).

md5sum

hashes it, and

```
awk '{print $1}'
```

makes sure there is only hash (and no extra garbage). And we write to file with operator.

Then we use hashcat to decrypt our pass_phrase like so:

```
$ hashcat -m 0 -a 3 0123.txt ?d?d?d?d --show
```

Where:

(a) -m specifies hash-type (in this case 0 which corresponds to md5),

(b) -a specifies attack type (3 corresponds to brute-force),

(c) we then pass input file number of characters (?d specifies that it is a digit) and

(d) (optionally) we use –show to show the answer when it is found.

Table 1: Time to crack the password made from digits.

#characters	time
4	0s
6	0s
7	0s
8	3s
10	6s
11	8min 11s
12	1h 24min

Table 2: Time to crack the password made from digits and small letters.

#characters	time
4	3s
6	7s
7	6min 20s
8	4h 39min

Later on we use also:

(a) -1 abcdefghijklmnopqrstuvwxyz0123456789 makes charset equal to small letters and numbers,

In task 1.3 in downloaded **10k most common passwords** and added **@437** at the end of it. When we specify that it is only 1 special character and 3 numbers results come in seconds (10s). When we don't know what is added at the end (but we know there is 4 of it) we may expect hashcat to work for couple of hours. We run **hashcat with -a 6 instead of -a 3** and we specify file with passwords as well as possible chars at the end.

```
$ hashcat –m 0 –a 6 letmein@437.txt 10k–most–common.
    txt ?a?d?d?d
```

In task 1.4 when we **don't use -m** flag hashcat will tell us which are the **most likely hash-functions** that were used to create that hash.

2 Zadanie 2

1. **Filtering options**: Wireshark allows filtering by **protocol, IP addresses, port numbers, and packet contents**.

2. **Types of data**: **HTTP** transmits **text-based** web content **(page content, cookies etc.)**, while **HTTPS encrypts it**. **SFTP** communication involves the transfer of files securely **over SSH**. Wireshark can capture packet headers, but the **file content** itself is **encrypted**.

3. **Usefulness in attacks**: Wireshark helps in

   (a) **Network reconnaissance**: Analyzing network traffic to identify potential vulnerabilities or targets.

   (b) **Malware analysis**: Identifying suspicious or malicious network activity indicative of malware infections.

   (c) **Data exfiltration**: Detecting unauthorized data transfers or suspicious outbound traffic.

   (d) **Intrusion detection**: Monitoring network traffic for signs of unauthorized access or abnormal behavior.

4. **Information from HTTP vs. HTTPS**: **HTTP** traffic **reveals URLs and content**, while **HTTPS encrypts it**. Wireshark can't easily reveal visited websites with HTTPS alone. However we can find such a data in http and especially in DNS by using:

   ```
   (dns.qry.name == "9gag.com") or (http.host == "9
       gag.com")
   ```

   This allows us to see if such a name appeared either in dns querries or in http host name. For "9gag" it only shows up in dns, but when we load some article from "wp.pl" their name is also visible in HTTP host name.