IT and Network Security
WiSe 2020/21

Practical Assignment No. 1

Nils Aschenbruck

Leonhard Brüggemann, Alexander Tessmer, **Stefanie Thieme**

Till Zimmermann

Release	27.10.2020
Submission	09.11.2020

General information for the practical assignment:

- Read the assignment sheet **thoroughly from top to bottom**, before you start working!
- For the admission to the exam, you have to acquire at least 8 points.
- You have to work on all assignments **on your own**.
- **(Code) plagiarism will not be tolerated!**
  This includes the use of code from other sources without reference or unusual amounts of code with references and applies to all parties involved.
- Submission is **no later than 9:00 a.m.** on the respective submission date via **StudIP**. Related files must be packed as a ZIP archive and uploaded in the assignment area of the lecture. The archive name must conform to the following pattern:
  ITS_WiSe_202021_PA1_*MatrNo*

  where ***MatrNo*** has to be replaced with your matriculation number. On unpacking the archive, a **subfolder with the same naming pattern** must be created automatically. Please note that potential changes to this submission method can occur for individual assignments.
- **Comply with all stated specifications** for file names, folder structure, program parameters, etc. Otherwise the time for assignment evaluation is unnecessarily extended and the result announcement will be delayed.
- Please use the **mailing list** of the lecture for questions of general interest. Alternatively you can attend the **Q&A tutoring session**. An application by mail to the teaching assistant (**thieme@uos.de**) is optional, but recommended for extensive questions.

Practical Assignment 1: Dictionary Attack (3 Points)

Modern web servers like the pervasive Apache web server [1] support a simple method to secure folders against unauthorized access: The usage of *.htaccess*-files (cf. [2]) and corresponding *.htpasswd*-files allows to enforce the input of user names and passwords from the HTTP client. While *.htaccess*-files are for configuration, *.htpasswd*-files store a combination of user names and hashed passwords. You already came across this concept of access restriction when you tried to download this very assignment sheet or the lecture slides: The HTTP client that is the web browser of your choice prompted you with a dialogue that asked you to enter your credentials.

Setup

We prepared a simple TCP login server (stupID_server.py) that secures the access to a secret similar to the above mentioned HTTP Basic Authentication. As the title of this assignment suggests, learning something about the HTTP protocol is not our main goal here. Therefore, you do not have to use any HTTP to solve the tasks – it is just about the concepts.

The stupID server accepts local connection requests (127.0.0.1, port 5000) of the following format:

```
<user name>:<base64-encoded SHA3 password hash>
```

and answers with an appropriate status message:

situation	status message
incorrect password	`00 - Password false.`
correct password	`01 - Password correct.`
connection aborted	`02 - Connection refused.`

The server aborts connections randomly to simulate disconnections. This can happen in real-world scenarios due to mobility of the client, overflow of queues or other networking problems.

Dictionary Attack

Brute force attacks are a naïve method to obtain access to hashed passwords like ours. The attacker simply tries every possible password until he is successful. As there exists a myriad of passwords, it is a good idea to reduce this number to a reasonable amount. We will take a look at one example: the dictionary attack. Since passwords are often made up of real words, an arbitrary text file (e.g. ASCII) can be parsed word-by-word in order to generate a dictionary. Your task is to write a python program that performs a dictionary attack on our login server.

By chance you know that there exists a user named *ITS202021*. Download RFC 4960 (text version)[1] to use as your dictionary. Within this context, a word is defined as a connected character sequence of arbitrary length without whitespaces. Assume that the password is protected by simple means against dictionary attacks. Therefore, each word has to be tested additionally with the following character replacement rules applied to all its characters:

Input:	o	i	r	e	a	s	g	t	b	p
	O	I	R	E	A	S	G	T	B	P
Output:	0	1	2	3	4	5	6	7	8	9

Your program call must look like this:

```
python3 pa1_client.py <path to dictionary>
```

Your program must output:

```
<password>: <user name>:SHA<base64-encoded SHA3 hash>
```

The part after the second colon corresponds to the format of a *.htpasswd*-file with a *SHA3-512*-hashed [3, 4] and *base64*-encoded [5] password.

In order to validate your obtained password, you can visit the following link:

`http://sys.cs.uos.de/lehre/its/2020/aufgaben2/test`

Protection Against Dictionary Attacks

Simple precautions can make it harder for a third party to perform a successful dictionary attack. Improve our server with an appropriate mechanism *beyond* the character substitution from above. If you find it hard to think of such modifications, maybe a quick web search can give you some inspiration...

```
"""
Please comment on the code changes that you have made and why that
helps to prevent an attacker from pulling off a successful
dictionary attack.
"""
```

Your modified server implementation must be named *less_stupID_server.py*

[1]`https://tools.ietf.org/rfc/rfc4960.txt`

Functional Check

Make sure that your programs terminate gracefully, even in the presence of connection failures, interrupts (e.g. SIGINT via CTRL-C) or the like.

Happy Coding!

Submission:

- ZIP-archive named *ITS_WiSe_202021_PA1_MatrNo.zip* that automatically creates the corresponding folder *ITS_WiSe_202021_PA1_MatrNo* on unpacking (cf. general information on page 1).
- This folder must contain all relevant source files (.py); at least:
  - the program that performs the dictionary attack: *pa1_client.py*
  - the extended server: *less_stupID_server.py*
- upload the zip-archive to **StudIP**: *IT- und Netzwerksicherheit* → *Tasks* → *PA1*

References

[1] `https://httpd.apache.org/`

[2] `http://wiki.selfhtml.org/wiki/Webserver/htaccess`

[3] `https://de.wikipedia.org/wiki/SHA-3`

[4] `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf`

[5] `https://en.wikipedia.org/wiki/Base64`