Attacks on Implementations of Secure Systems

Chen Doytshman, Igor Alikin, Tomer Wasserman

Assignment 1 Report - Temporal Side Channel

1 Sample Executions

Below are sample executions that were made using our script. The not-yet-recovered characters are printed as asterisks (`*`). The printouts were made to `stderr`, while only the final password recovered was made to `stdout`. Please note: we show here an execution to the highest difficulty we achieved, which took less than 6 hours. According to Yossi, mentioning the highest difficulty we solved in under a minute is not necessary.

Command Line

```
$ python3 ex01_M2.py 205644941 14
Sending wakeup requests...
Getting password length...
Password length is 16
---------------------------------
Current password: ****************
Guessing character at index 0
Best guess for index 0: h
---------------------------------
Current password: h***************
Guessing character at index 1
Best guess for index 1: p
---------------------------------
...
---------------------------------
Current password: hpwuzjimilwqco**
Guessing character at index 14
Best guess for index 14: a
---------------------------------
Current password: hpwuzjimilwqcoa*
Brute-forcing last character...
Best guess for index 15: k
---------------------------------
hpwuzjimilwqcoak
Finished in 272.31 minutes
```

Command Line

```
$ python3 ex01_M2.py 322081241 14
Sending wakeup requests...
Getting password length...
Password length is 16
---------------------------------
Current password: ****************
Guessing character at index 0
Best guess for index 0: l
---------------------------------
Current password: l***************
Guessing character at index 1
Best guess for index 1: l
---------------------------------
...
---------------------------------
Current password: llqoraayiwvpsh**
Guessing character at index 14
Best guess for index 14: t
---------------------------------
Current password: llqoraayiwvpsht*
Brute-forcing last character...
Best guess for index 15: n
---------------------------------
llqoraayiwvpshtn
Finished in 289.22 minutes
```

Below samples were taken with usernames instead of IDs, before our forum question was answered.

More samples:

`python ex01_M2` chendoy 13 → zfncsbyohbtvbuta
`python ex01_M2` chendoy 12 → bbetcexaagdcykpc
`python ex01_M2` chendoy 11 → basxnznmufpiyedn
`python ex01_M2` chendoy 10 → jbgtaxtxsrijalql
`python ex01_M2` tomerwa → adalya

In order to be as flexible as possible, we have used different parameters configuration for each difficulty we have solved. It allows using subtle configuration in lower difficulties and more aggressive ones in higher difficulties.

> **Dear grader:** As written to you in the mail, Tomer Wasserman has joined the group near the submission deadline, by a request from the course staff. He joined when server load was heavy, Therefore, no difficult password could be discovered for him to this moment, even from within the university network. We would appreciate your consideration regarding this issue.

2 Analysis

With a brute-force approach and for a known password length and for characters set a-z it would require us to make 26^N requests to the server, when N is the password length. Since the password length is unknown to the attacker, we will have to accumulate the former term over N, yielding maximum number of $\sum_{i=1}^{32} 26^i$ attempts. In asymptotic notation it is considered $\mathcal{O}(26^{32})\approx$O($2^{150}$).

With our program, which exploits the temporal side channel present in the server, we first guess the password length. Then, proceeding with some password length, we try to crack the password, one character at a time. Finally, at the last character we can use "brute-force" since the server response is now usable. The number of attempts we do now can be written as:

$$\mathcal{O}(B \cdot N + C \cdot n \cdot \sum_{i=1}^{\log_{1/D} Z-1} 2^i + Z) = \mathcal{O}(C \cdot n \cdot 2^{log_2 Z-1+1} + Z + C \cdot N) = \mathcal{O}(C \cdot N \cdot (Z+1) + Z)$$

When:
B - the number of requests sent per measure (length).
C - the number of requests sent per measure (characters).
D (THRESH)- the quantile of lowest measurements to remove before next measurement.
Z - the size of the set of possible characters.
n - the determined password length.

In the above equation we took $B = C$ to get a cleaner term. When considering 16 characters long password (difficulty 14), the number of requests reduces to $9 \cdot 32 \cdot 27 + 26 = 7802 < 2^{13} \ll 26^{16} \approx 2^{75}$.

3 Optimizations

We have used various optimizations in our code, some of which were used for speed while others are for robustness (i.e recovering the correct password). The trade-off between these optimizations is obvious: one can apply very strict speed optimizations and pay with low recovering rates, while other can apply a very conservative approach and pay with a very slow program. In order to deal with differences between our home network and the university network we used more robust parameters for our script (e.g more sampling per measure, smaller quantile dropout etc), this is due to additional noise that can be present when working with the university VPN.

3.1 Lowest Quantile Dropout

When measuring times for the i'th character, we maintain a dictionary that maps characters to their accumulated times (and chances count, see next optimization). After each iteration we remove from the dictionary the `FILT_THRESH` characters with the lowest measurement. This allows us to proceed to the next measurement only with the characters that are more likely to be the correct one, thus deceasing times. For `FILT_THRESH=0.25`, the idea behind this optimization is that if some character is the next one and takes more time than the others, it is not likely to be in the 25% most shortest measurements.

3.2 Multiple Chances per Character

In order to increase the robustness of our program to noise, especially in the higher difficulties, we have added multiple chances for character before dropping them out. The idea behind this optimization is as follows: having our previous optimization, we would fail if we accidentally drop the correct character instead of picking it. This would happen if this character would happen to be in the FILT_THRESH lowest measurements in some iteration. Since this can happen by noise, we give "second chance" (and third, forth, and so on..) to characters. In other words, we actually drop a character if it happens to be in the FILT_THRESH lowest measurements for NUM_CHANCES rounds.

3.3 Reusable HTTP Session Object

Moving to the technical side of things, we wanted to make our code free-of-noises as possible. One way to to so is to remove time-consuming network operations that blurs always the times we are interested in - the time that it takes to the micro-processor to execute the server-side code. To this end, we have used python's persistent HTTP `Session()` object. It maintains a pool of connections, allows reusing an already opened connection(s), and enables reduced latency in subsequent requests (no TCP handshaking).

3.4 Differences Between Networks

To deal with differences between the university network and our home network behind a VPN, we basically used more robust parameters. For example: we sampled each letter 5 times instead of 3, to cope with the additional noise. In addition, we used the requests library built in `res.elapsed.total_seconds()`instead of measuring time with python's `time.time()`.

4 Appendix: difficulties parameters

Below are the parameters we have used for each difficulty we solved.

```
 [
{
   "DIFFICULTY":0,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":1,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":1,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":1,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":2,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":1,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":3,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":1,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":4,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":3,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":5,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":3,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":6,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":5,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":7,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":5,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":1,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.5
},
{
   "DIFFICULTY":8,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":7,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":2,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.4
},
{
   "DIFFICULTY":9,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":7,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":2,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.4
},
{
   "DIFFICULTY":10,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":7,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":2,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.4
},
{
   "DIFFICULTY":11,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":9,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":3,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.25
},
{
   "DIFFICULTY":12,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":9,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":3,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.25
},
{
   "DIFFICULTY":13,
   "NUM_ROUNDS_LENGTH_GUESS":16,
   "NUM_ROUNDS_CHARS_GUESS":11,
   "MAX_PASSWD_LENGTH":32,
   "NUM_CHANCES":4,
   "TIMEOUT":20,
   "RETRY_LIMIT":10,
   "FILT_THRESH":0.2
},
{
      "DIFFICULTY":14,
      "NUM_ROUNDS_LENGTH_GUESS":16,
      "NUM_ROUNDS_CHARS_GUESS":11,
      "MAX_PASSWD_LENGTH":32,
      "NUM_CHANCES":4,
      "TIMEOUT":20,
      "RETRY_LIMIT":10,
      "FILT_THRESH":0.2
   }
]
```