Pass the Hash

Warmup/Learning, 50 points
Author: rtitiu

In this task the hash function $\mathsf{combo_{hash}}()$ produces an output from a salt and a password by mixing together four secure hashes. A secret 20 bytes password is generated and we are given oracle access to hashes computed using this password and salts that are under our control. At the end of this query phase we are given a uniformly random salt and if we are able to generate the corresponding hash we get the flag. In order to do this we will actually recover the secret password.

To start, let's assume that the hash function $\mathsf{combo_{hash}}()$ only uses secure hash functions that output 20 bytes (for instance $\mathsf{sha256}$ outputs 32 bytes, so this would not be allowed).

Let's denote

$$L^0_{pass}||R^0_{pass} := \mathsf{password}||\mathsf{salt}||\mathsf{password}$$

and $L^i_{pass}||R^i_{pass}$ with $i \in [1, 16]$ the hash value at round i. So $L^{16}_{pass}||R^{16}_{pass}$ is actually the output of the $\mathsf{combo_{hash}}()$. Notice that what happens during the computation is:

$$L^i_{pass} = \mathsf{xor}\left(L^{i-1}_{pass}, h_{i-1}(R^{i-1}_{pass})\right)$$

$$R^i_{pass} = \mathsf{xor}\left(R^{i-1}_{pass}, h_{i-1}(L^{i-1}_{pass})\right)$$

for all $i \in [1, 16]$ and h_{i-1} is one of the secure cryptographic hash functions. By looking at the output of the hash value we notice that L^{16}_{pass} is equal to the initial value L^0_{pass} xor-ed with a 20 bytes string $\mathbf{w}$ (this is because we assume that every secure hash outputs only 20 bytes). Remember that $L^0_{pass} = \mathsf{password}||\mathsf{salt}[:12]$. Also notice that the xor function from the script repeats the second string of the input, namely $\mathsf{xor}\left(L^0_{pass}, \mathbf{w}\right) = \mathsf{password} \oplus \mathbf{w}||\mathsf{salt}[:12] \oplus \mathbf{w}[:12]$, thus we can actually recover $\mathsf{password}[:12]$ since we know the salt value. By working with the right half of the hash we can recover $\mathsf{password}[-12:]$ in the same manner. Thus we would be able to recover the password.

In order to solve the task, notice that the probability that $\mathsf{combo_{hash}}$ uses only secure hashes with output length 20 is equal to $(3/4)^{16} \approx 0.01$. The probability has this value since 3 out of the 4 secure hashes output 20 bytes (all of them except $\mathsf{sha256}$) and we run the same process for 16 rounds. This roughly means that for every 100 queries $\mathsf{combo_{hash}}()$ uses only 20 bytes long secure hashes. Thus we have reduced our original problem to the previously discussed case.

There is still one problem that we have to solve: how do we know when the output for a query was computed using only 20 bytes long secure hashes? For each query that we make we compute a password candidate as

$$\mathsf{password}[:12] := \mathsf{salt}[:12] \oplus L^{16}_{pass}[:12] \oplus L^{16}_{pass}[20:32]$$

as described above (as well as $\mathsf{password}[-12:]$). We can check that the overlapping bytes from $\mathsf{password}[:12]$ and $\mathsf{password}[-12:]$ are the same. With high probability, this happens only when all the secure hashes output 20 bytes.