CS177: Computer Security
Prof. Stefano Tessaro

UC Santa Barbara
Fall 2018

Homework 3

Posted: Wednesday, October 17, 2018 – 11:59pm
Due: Friday, October 26, 2018 – 11:59pm (Gradescope)

Task 1 – Modular Arithmetic and RSA Encryption (7 points)

a) [Points: 2] What is $\phi(2214119)$?

Note: If you use a small program to find this out, you do not need to submit it, but do explain how you got there!

b) [Points: 3] Assume that we use the RSA public key with $N = 2214119$ and public exponent $e = 5$. What is the corresponding decryption exponent d?

Again, explain your answer!

c) [Points: 2] We have seen a construction in class of a full-fledged RSA-based randomized cryptosystem following the PKCS#1 standard or RSA-OEAP. Alice proposes a different way to introduce randomization into RSA encryption: Given an RSA modulus $N = P \cdot Q$ and a valid public exponent e, to encrypt a message $m \in \mathbf{Z}_N$, we first choose a random r from $\mathbf{Z}_N$ and the ciphertext as $c = (m \cdot r)^e \bmod N$.

Is this a good idea? Explain!

Task 2 – Digital Certificates (4 points)

Consider the setting used to present key-exchange in class, where the client holds the verification key VK^* of the certification authority (CA). In particular:

> **(1)** The client communicates to the server `domain.com` the intention to open a secure channel.
>
> **(2)** The server at `domain.com` responds with a certificate $\text{cert} = [\mathsf{PK}, \texttt{domain.com}, S]$, where S is a signature (under the CA's signing key SK^*) of $[\mathsf{PK}, \texttt{domain.com}]$.
>
> **(3)** The client checks that S is valid by running $\text{Verify}(\mathsf{VK}^*, ([\mathsf{PK}, \texttt{domain.com}], S))$, and if valid, it generates a key K using AE-Kg, and sends $C \leftarrow \text{PKE-Enc}(\mathsf{PK}, K)$ back to the server.
>
> **(4)** The server decrypts C to obtain K, and both client and server are going to use K.

a) [Points: 2] Assume that a person-in-the-middle attacker can intercept and alter the entire communication between server and client. Explain how such an attacker can force the server to accept a key $K' \neq K$, known to the attacker, and different from the one sent by the client.

b) **[Points: 2]** Explain why the attack from **a)** is not really a problem from the perspective of the server.

Hint: What kind of authenticity guarantees can the server expect at the end of this protocol?

Task 3 – Crypto Audit for Password Hashing (4 points)

Download the Python code at `https://www.cs.ucsb.edu/~tessaro/cs177/hw/phash.py`. The procedure `hash` takes as input an arbitrary password string (or phrase), and produces a hash of length 256 bits, obtained as the concatenation of two hashes produced by the MD5 hash function.

Inspect the code, and describe in detail at least *four* vulnerabilities that make `hash` unsuitable for password hashing. Also, briefly explain how each vulnerability could be exploited.

Task 4 – Password Cracking (20 points)

The goal of this task is to crack the passwords of four accounts on a dedicated machine. Several password cracking tools are available, most of them for free – however, you will only learn a limited amount of skills by using those. This homework is about implementing *your own password cracking* tool (in Python or in C).

The accounts. We have made available on the Piazza resource page a portion of the corresponding `/etc/shadow` file containing usernames and password hashes, together with the IP address of the machine. The passwords have different degrees of difficulties, and the hashes are computed with different hashing algorithms. You may want to use different strategies for cracking – brute force search, dictionary attacks, intelligent guessing, using common substitution patterns, and combinations thereof. Be creative! *There is no "right" or "wrong" approach.*

Submission. You are required to submit your solutions to gradescope. You will see two assignments on gradescope: **Homework 3** and **Homework 3 Code**. Submit your solutions as follows (read the following carefully!).

1. Submit code (Python or C) of your password cracker, with clearly understandable comments to **Homework 3 Code**. Please be careful to upload your code to the **Homework 3 Code** assignment, and *not to Homework 3*, which is where you will upload your solutions to problems 1-3.

2. *The TAs must be able to compile your code on the CSIL machines*, so make sure to include all necessary files! Uncompilable code will result in 0 points being assigned.

3. Submit to **Homework 3** (along with your solutions to problems 1-3) an explanation of what you tried and what failed in detail. You may obtain most of the points even if you dont recover all passwords, as long as your explanation justifies sufficient

effort on your part. Also, if you succeed in recovering passwords, indicate benchmarking information on how fast the process went. Once again, your explanation needs to be submitted along with your solutions to problem 1-3 to the **Homework 3** assignment on gradescope. The **Homework 3 Code** assignment is *only for code submission!*

4. **Do not submit the passwords in any form or include them in your explanation.** If you recovered a password, log into the corresponding account and create an empty file named FirstnameLastname in the directory Visitors, e.g., by executing touch Visitors/FirstnameLastname, where obviously Firstname and Lastname should be set accordingly.

Available resources. To help you out with the task, we are sharing with you the following resources on our Piazza page:

- The actual `/etc/shadow` file, with the username and the encrypted passwords.
- Examples in C and Python of toy programs hashing one password with the corresponding salt. These are named `crypt_example.c` and `crypt_example.py`. You can use these as a starting point for your password cracker. A post on Piazza already explains in detail how to use them and compile them.
- A dictionary of English words.

Some rules. This homework requires us to access existing valid accounts on an Internet-connected machine and to perform fairly intensive computations. In order for this type of homework to remain possible, it is crucial that you stick to the rules:

- **You only have permission to break into these four accounts.** If you do succeed in gaining access, *do not* make any modifications to the accounts, change passwords, run and install software. Just create the empty file as explained above.
- **You are not allowed to run an *online* attack on the accounts or any other form of attack other than a password-cracking offline attack.** You will know if you found the correct password, so there is no reason to flood our server with invalid logins. Be considerate.
- **You are not allowed to waste resources on CSIL machines.** Limit yourself to running your password cracker on *one single* CSIL machine. If additional computing power is needed, you have to rely on your own personal devices.
- **You are not allowed to run any password cracking computations on the remote login server** `csil.cs.ucsb.edu`**.** Any other machine listed at `http://www.engr.ucsb.edu/eci/kb/index.php?action=artikel&cat=14&id=68` is acceptable.
- You are not allowed to share the `/etc/shadow` file with anyone, and you are not allowed to share the passwords (if found) with anyone else, nor send them via e-mail, nor write them down.

Final hints. Make sure you understand the `/etc/shadow` format and the usage of `crypt`. In particular, the cracking should occur on Linux, though other free UNIX systems may do just fine too. Mac OS X is not a good option.

Also note that while some passwords have weaknesses, we still needed to comply with certain regulations (all passwords are at least 6 characters), so even the easiest password will take some time to be recovered.