EITA25 - Lab 1

André Frisk & Johan Åkerman

February 17, 2022

Preparatory Questions

1.
   - **What is a brute force attack?**: In a brute force attack, a threat actor tries to gain access to sensitive data and systems by systematically trying as many combinations of usernames and guessed passwords as possible. If successful, the actor can enter the system masquerading as the legitimate user and remain inside until they are detected. They use this time to move laterally, install back doors, gain knowledge about the system to use in future attacks, and, of course, steal data.
   - **What is a dictionary attack?**: A dictionary attack is a method of breaking into a password-protected computer, network or other IT resource by systematically entering every word in a dictionary as a password. A dictionary attack can also be used in an attempt to find the key necessary to decrypt an encrypted message or document.
   - **What is a time memory tradeoff attack?**: Time-Memory Trade-Off (TMTO) attacks offer a generic technique to reverse one-way functions, where one can trade off time and memory costs and which are especially effective against stream ciphers. This attack is a special version of the general cryptanalytic time/memory tradeoff attack, which has two main phases: **Preprocessing**: During this phase, the attacker explores the structure of the cryptosystem and is allowed to record their findings in large tables. This can take a long time. **Realtime**: In this phase, the cryptanalyst is granted real data obtained from a specific unknown key. They then try to use this data with the precomputed table from the preprocessing phase to find the particular key in as little time as possible.
   - **What is the difference between a rainbow table and an ordinary time memory tradeoff table?**: Time memory tradeoff classic tables is of size m x t while one rainbow table is of size mt x t. It takes half as many operations to look up a key in a rainbow table than in t classic tables.
2.
   - **How is the LAN Manager (LM) hash produced, how is this hash used to authenticate a user, and what is the effective security of LM?**: LM works by creating a "hash" of your password, as follows: **Breaking the password into seven-character chunks**: If the password length is not a multiple of seven (i.e., 7, 14, 21, 28...), LM pads the remainder of each chunk with zeroes to make it a full seven bytes long. For example, a ten-letter-long password would be divided into a seven-character-long chunk and a three-character-long one, and the smaller one would have zeroes added to it to push it up to seven characters. **Mapping all lowercase letters into uppercase**: For example, Dog becomes DOG. **Encrypting each chunk**: LM uses each chunk as a 56-bit DES (a standardized cipher standard) key to encrypt the following string: KGS!@#$%. **Concatenating those strings (i.e., linking them**

end-to-end). The resulting string is what computers pass between each other when authenticating users with the LM protocol.

In terms of security, this is the lowest level at which any Windows computer can operate. Windows LAN Manager (LM) hashes are known to be weak. The LM hash has several weaknesses in its design. This makes such hashes crackable in a matter of seconds using rainbow tables, or in a few minutes using brute force. Starting with Windows NT, it was replaced by NTLM, which is still vulnerable to rainbow tables, and brute force attacks unless long, unpredictable passwords are used. **The biggest vulnerabilities for LM is that the password length is limited, password are not case sensitive, the dividing of the 14-character password makes it easier to crack, the hash value is sent to network servers without salting which makes man-in-the-middle attacks and time-memory tradeoff attacks possible**.

- **How is the NT LAN Manager version 1 (NTLM) hash produced, how is this hash used to authenticate a user, and what is the effective security of NTLM?**: NTLM authentication typically follows the following step-by-step process: The user shares their username, password and domain name with the client. The client develops a scrambled version of the password — or hash — and deletes the full password. The client passes a plain text version of the username to the relevant server. The server replies to the client with a challenge, which is a 16-byte random number. In response, the client sends the challenge encrypted by the hash of the user's password. The server then sends the challenge, response and username to the domain controller (DC). The DC retrieves the user's password from the database and uses it to encrypt the challenge. The DC then compares the encrypted challenge and client response. If these two pieces match, then the user is authenticated and access is granted.

  NTLM is generally considered insecure because it uses outdated cryptography that is vulnerable to several modes of attacks. NTLM is also vulnerable to the pass-the-hash attack and brute-force attacks.

3. - **Ensure that you understand what a salt is, and why it is used**: In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes data, a password or passphrase. Salts are used to safeguard passwords in storage, used to protect against duplicate or common passwords being identifiable (as their hashes are identical). A new salt is randomly generated for each password. Typically, the salt and the password are concatenated and fed to a cryptographic hash function, and the output hash value (but not the original password) is stored with the salt in a database. Salts don't need to be encrypted or stored separately from the hashed password itself, because even if an attacker has access to the database with the hash values and the salts, the correct use of said salts will hinder common attacks.

   - **Find information and read about PBKDF2. Roughly compare the time required to calculate a hash with PBKDF2 with the time required for a single round of SHA-1. You do not have to understand the algorithm in detail**: PBKDF2 applies a pseudorandom function, such as hash-based message authentication code (HMAC), to the input password or passphrase along with a salt value and repeats the process many times to produce a derived key, which can then be used as a cryptographic key in subsequent operations. The added computational

work makes password cracking much more difficult, and is known as key stretching.

PBKDF2 are configured with an "iteration count", which means that they can be made as slow as you want. Therefore, there cannot be a table which shows how fast they go. What you need to do, instead, is to decide how much time you allocate to the function (e.g. you want it to take 0.05 seconds on your server) and then set the iteration count accordingly. This very much depends on the computing abilities of your server, so this is a matter of measure on your target hardware.

SHA-1 processes the data by chunks of 64 bytes. The CPU time needed to hash a file of length n bytes is thus roughly equal to n/64 times the CPU time needed to process one chunk.

4.
   - **Look up the meaning of the term Penetration testing if it is unfamiliar to you**: A penetration test, colloquially known as a pen test or ethical hacking, is an authorized simulated cyberattack on a computer system, performed to evaluate the security of the system. This is not to be confused with a vulnerability assessment. The test is performed to identify weaknesses (also referred to as vulnerabilities), including the potential for unauthorized parties to gain access to the system's features and data, as well as strengths, enabling a full risk assessment to be completed.
   - **Read this short security bulletin about one of the exploits we will use during the attack: https://helpx.adobe.com/security/products/flash-player/apsa15-03.html**: A critical vulnerability (CVE-2015-5119) has been identified in Adobe Flash Player 18.0.0.194 and earlier versions for Windows, Macintosh and Linux. Successful exploitation could cause a crash and potentially allow an attacker to take control of the affected system.
   - **Glance through the possibilities of the Metasploit's Meterpreter, which we will use https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics**/: See the guide for commands and how to use Meterpreter.

Lab Questions

1. **What passwords can you find in Firefox? (Write down at least one)**: EITA25rUleZ!!1
2. **What passwords can you find in Chrome? (Write down at least one)**: Qval13t!
3. **What is the difference between the two browsers? Does any browser handle the passwords more securely than the other? Explain!**: Chrome demands a password input of the user to access the hidden password, FireFox does not.
4. **Why don't they hash the passwords?**: Otherwise we can't read them and know what our password is.
5. **Find the passwords of your chosen accounts**: Labuser1a: dictionary does not work, 0 / 2 hashes. Rainbow gave us I8ZZAHM and 3JK1337.

   Labuser1b: dictionary gave us NQPCAN and PR09GF2. Rainbow gave us the same.

   Labuser2a: dictionary gave us continentally23 for dictionary. Found no tables for rainbow.
6. **Which password(s) can be found by dictionary attacks? Which password(s) can be found using the rainbow table. Explain the results**: Labuser1b & 2a worked with dictionary attack. Labuser1b worked with rainbow table.

7. **Give a password that would not be possible to crack in this way**: Not use dictionary words and add salt to make rainbow tables impractical.
8. **How long would it take to try all passwords using brute-force? Remember to set the correct length and character set inside Cain before starting**: Estimated 7 hours.
9. **Launch a brute-force attack on any of the MS-CACHEv2 hashes you find. Use the same character set and password length as before. How long time would it take to test all passwords? You do not have to wait for it to finish**: Estimated 1200 years.
10. **Is the salt well chosen in Windows? Why or why not?**: No, it uses the username as salt.
11. **Which users and groups have administrative access to the machine?**: Administrator, COMPSEC/CompsecUser, COMPSEC/Domain Admins, IEUser.
12. **Who owns the directory Alice? Who can (according to the ACL) access the directory and what can they do?**: Alice owns it. Admin (full control), system (full control), users (read and execute). Authenticated users (modify).
13. **Who can (according to the share permissions) access the directory and what can they do?**: Everyone (read).
14. **Can you access the file? What rule does Windows 10 apply here?**: No, because Alice is included in Everyone.
15. **Give Bob Full control to the file. Can Bob read the file? Can he write to the file? Can he change the permissions on the file? Can he take ownership of the file?**: Can read, write, change permission and take ownership.
16. **Can you write to the file and can Bob write to the file? Experiment with different permissions for the folder/file and the sharing permissions. What rules are used here?**: I can write to the file. Bob can do it too. Sharing permission: everyone can read. Security: everyone full control.
17. **Which user is currently logged in? (Use the command whoami)**: ieuser
18. **Write down the privileges you have? (Use the command whoami /priv)**: sechangenotifyprivilege (bypass traverse checking).
19. **Where is the file hacked.txt located? write down the path and go to the path with cd [path] command.)**:

    C:/users/IEUSer
20. **What is the content of the file? (Use the command type hacked.txt to read the content)**:

    "You have been hacked"
21. **What are the processes running on target? (Use the command tasklist to see the processes, you need to wait for few seconds to see all running processes.)**:

    see console.

22. **Try to kill the process named registry, were you able to kill this process? why or why not? (Use the command taskkill /PID XX /F to kill th process, replace XX with the PID of registry process that you found in previous step.)**:

    No, access denied!