PASSWORD DESIGN & CRACKING

# lesson

**Lesson:**

* You will need to install the password cracking tool call John the Ripper. Open a bash terminal and run the following command:

**sudo apt-get install john**

**y**

**enter**

* Explore GRC Password Haystacks & discuss attacks on passwords
  + https://www.grc.com/haystack.htm
  + The goal of setting a good password/phrase is to force the attacker to use brute force instead of a cracking dictionary/wordlist (one layer of **defense in depth)**
  + Once you get to brute force attacks longer passphrases that contain a larger character set will make things exponentially more difficult
  + Some sophisticated cracking programs use morphing to try common modifications to words/phrases in a dictionary such as adding numbers onto the end, replacing certain characters with numbers that look similar, etc.
  + When trying to crack a password there is no feedback other than if the attempt was successful or unsuccessful, attackers have no way of knowing if they are close or not. There is no “locking in of characters one at a time” like is commonly seen in movies and TV, you need to guess the whole password at once and then find out if that was correct or not. (**thinking like an adversary**)
* Understanding how passwords are stored – gives you an idea of how/why attackers will attack in certain ways (**thinking like an adversary**)
  + In well designed databases passwords are not stored in plain-text (sometimes called cleartext) but instead a **hash** of the password is stored along with the username.
  + This prevents someone who obtains a copy of the database from directly reading the passwords of all the users.
  + Hashing is a one-way cryptographic function. This means it cannot be undone like encryption can, but it’s still very useful for comparing things like passwords to store them securely in a database without them being readable but still be able to tell if what someone enters later matches what they have set their password to. Hashing can also be used to verify the **integrity** of data (that it has remained unaltered).
* Try Password Dictionaries & Brute Force Utilities
  + Install “john” and “john-data” john the ripper software packages
    - These tools conduct a real offline attack of the hashed passwords in the user database file you supply.
  + Create a new file in your home directory named “pwds-to-crack”
    - Open the file with a text editor (open a bash terminal and type mousepad and hit enter)
    - The format of this file should be:
      * *username*:*hashed\_password*
      * One user/password hash combination per line
      * Make up any username you want, these are just for demonstration
      * Get the hash of the password by running **openssl passwd -1 *password\_here***
        + That’s a number 1 not a lowercase L in the command line
        + Copy and paste the result into the file in the *username*:*hashed\_password* format
      * A screenshot of a computer

        Description automatically generatedSuggest creating several password hashes in the file starting from easy, short dictionary based words like dog, cat, etc., to harder/longer ones
  + Hit File…save…and name this file ‘pwds-to-crack’ and hit save
  + Back in the terminal, run **john pwds-to-crack** to try using a built in dictionary first and then progressing on to a brute force attack
    - Press a key while john is working to get statistics about the cracking
  + If you want to force john to use brute force and ignore the built-in dictionary run **john --incremental *pwds-to-crack***
  + If you want to run john again on the same passwords you need to delete the **~/.john/john.pot** file between runs, otherwise it will use the stored pre-cracked passwords from the previous run
  + Try using <https://crackstation.net> to quickly crack password hashes using a very large dictionary.
    - Use **echo -n “*password*” | sha256sum** to create the values to paste in the Crackstation site, these are SHA-256 hashed passwords.
    - This site demonstrates a *rainbow table*attack, this means the dictionary is pre-hashed and stored in a database making it easy, quick, and much less CPU intensive to crack passwords.
    - Well-designed password storage systems prevent the use of pre-hashed rainbow tables by using *salt* in their hashes, automatically adding something random and unique per-site (ok) or per-user (much better) to the end of each user’s password before hashing it. This means you would need to create a custom rainbow table for each site or user which defeats the speed advantage of having a rainbow table in the first place.
  + Now try cracking the passwords in the following two files found in the Thursday github folder:
    - FourMD5PasswordHashes.tx
    - GenCyberPasswords
    - Camperpasswords

Helpful hint

* If you want to tell john what to use a rainbow table (wordlist), you need specify a wordlist. Try the below to see how this command works:

John pwds-to-crack -w=wordlist

Rockyou.txt is a common wordlist that should be in your Thursday folder.