Password Cracking Worksheet

Summary: Web applications must not store passwords. They are allowed to store an irreversible, salted hash of the password. When a pen tester acquires these hashes, it is a major victory but the work is not done. Password "cracking" tools are needed to determine what password was used to make the hash.

Assignment: Given the following clues, use John the Ripper (JTR) to "crack" the following sets of passwords

1. JTR is not installed by default on Samurai but the default version can be installed easily
    a. root@samuraiwtf:~# cd /opt
    b. root@samuraiwtf:/opt# mkdir john
    c. root@samuraiwtf:/opt# cd john/
    d. root@samuraiwtf:/opt/john# tar xvzf john-1.8.0-jumbo-1.tar.gz
    e. root@samuraiwtf:/opt/john # cd /john-1.8.0-jumbo-1/src
    f. root@samuraiwtf:/opt/john # ./configure
    g. root@samuraiwtf:/opt/john # make
    h. root@samuraiwtf:/opt/john/john-1.8.0-jumbo-1/src# cd ../run/
    i. root@samuraiwtf:/opt/john/john-1.8.0-jumbo-1/run# ./john

2. **Clear Text Passwords**: Sometimes apps fail to hash passwords instead storing the actual password in clear text. Attack the user-info.php page in Mutillidae with SQL Injection and extract the passwords. List the passwords in the following format: <username>:<password>. Example: jeremy:Password1. **Tip**: Rumor has it someone posted a video on the Webpwnized YouTube channel that may be useful.

3. **Obfuscation**: Developers may confuse encryption with obfuscation. While the following passwords may appear encrypted, they are not. John the Ripper would be serious overkill.
    a. "Crack" the passwords
    b. What is the encoding used to "encrypt" the passwords?
    c. What tool in Burp-Suite is useful in this scenario?

    Password1
    123456
    ThisPasswordIsNotBad998#$
    @Diff9cult2Guess@
    ItDoesNotMatterHowGoodThePasswordIfThePasswordIsNotHashedProperly*#&@
    %$$44

    1. UGFzd29yZDE=
    2. MTIzNDU2
    3. VGhpc1Bhc3N3b3JkSXNOb3RCYWQ5OTgjJA==
    4. QERpZmY5Y3VsdDJHdWVzc0A=
    5. SXREb2VzTm90TWF0dGVySG93R29vZFRoZVBhc3N3b3JkSWZUaGVQYXNzd29yZEl
    zTm90SGFzaGVkUHJvcGVybHkqIyZAJSQkNDQ=

4. **Hashing without salts**: Crack these passwords using JTR. Perhaps the default wordlist can crack the passwords but most likely an alternate list may be needed. For each set, show the clear text password next to the hash. Also identify what hashing algorithm was used to hash the set. Within a given set, the hash algorithm used is the same.

   Example Answer: 5adfaa8264caf8e271b455072e37af3e redwings (MD5)

   Set 1:

   samurai@samuraiwtf:~$ for i in `cat /tmp/wordlist`; do echo -n $i | md5sum; done;

   redwings
   rosebud
   rush2112
   saturn
   scooby
   scorpion

   60ccc193cb458437b29698fad4ba2e23
   09bb63bf7635f9cfd50f022e7a3b0dba
   f71f21a84e7fec0da740b689c7b0bb8e
   d027eb0ee23c9fcaa2b9ce4f221c5a77
   b0fc08a18d29407428cbac5d2e5cc682
   3f8da8d150df71f64db5f8e96438c567

   Set 2:

   for i in `cat /tmp/hashes`; do echo -n $i | sha1sum ; done;

   thx1138
   tigers
   trustno1
   tucker
   twitter

   568b156009ca4316b0d656da88f0e1c2aceb2185
   263d00820f9f5e0acc0274da747e0a9b6868145e
   e68e11be8b70e435c65aef8ba9798ff7775c361e
   014a5f52613b4742a930f7f953ee9f59bdd19769
   8a1621dae39bf1d91d372c77f441e80b8f68b9b6

   Set 3:

   for i in `cat /tmp/wordlist`; do echo -n $i | sha512sum ; done;

ncc1701
newyork
nicholas
oliver
orange

a. 4b45e6839e5d65ebce1c64343ca80543344678870aaf9bbbf8126b6c062e75313112ff79a944429d1a97fc4e0cc58bfa695ca8daf3ed900923988f2068b75449
b. 8512de11f6042ae4128256c8e6c1bfb68ee50434ab09ae0973cba0c09557619383f4ca6dfdb7673156ee21502ba6412d3f8b94b327b9438ecc24f57b57dc3afc
c. 4dfc4f3ca31220a70bfa9ea196fb73cefa0d61f664bd73cbfae8080633e984644c31522101a39c93f076e4d489c89988181dd46290f50ab4a47a0c675eb724cb
d. b4e6749fe6bc93783b33a14a4b1e795f71fd66ace13ff7e6e5bffe86337bbedbc37ecfeee8a355421e88d76e1e1b9aaf53e7e95152ba39a2961d5a1f4b3d39c2
e. 84fe251777706fc59b72b34bae43eb4c691d3aed700a85990e3a0224ac8e01efe3984429b99fc4a209c7e971f7e2916bf488d9500553f7f78c2d5ea23d0fa558

5. **Hashing with salts**: Crack these passwords using JTR. Perhaps the default wordlist can crack the passwords but alternates may be needed. For each set, show the clear text password next to the hash. Also identify what hashing algorithm was used to hash the set.

   Set 4:

   al:twitter5:1001:1001:Test Account:/home/al:
   bill:yellow3:1002:1002:Test Account:/home/bill:
   charley:welcome7:1003:1003:Test Account:/home/charley:

   a. al:6JbKaru3X$CEvI80ugA.IkChkeUpHiI/BkJxC8ZLPpJXHIIlfO2/mUpCRc23f.zZnaiDqGQmJJjtrtXldSGumfGyX/xqDUu/:1001:1001:Test Account:/home/al:
   b. bill:6OeCU47gD$1Jt1StdqyEvVkK0NhMRiVsT4LEUYYOyDDFDAtaKr5P.FcwzaZwuLBXgz8WDxoTmAcM7jcDto1/QE3YIS8p/qM.:1002:1002:Test Account:/home/bill:
   c. charley:6Z0Ko3Kxz$f1HNHq7NxhZMLwlzsbdiai4pWGqTLJkgsahyTqjVYpzPh9jgxt0g/26.JjekFssF1/5dNj2RbW8oa.OCglFQk0:1003:1003:Test Account:/home/charley:

Tips:

a. JTR help can be obtained at the command line by typing “john”

b. Using wordlists from various sources can be helpful as wordlists can be kept in the pen testers library for specific scenarios and combined to build dictionaries. Samurai and Kali Linux contain many wordlists that can be found using “locate wordlist”. The Skull Security site has collections of passwords that are free to download. Note this project only requires wordlists found in Samurai.
   a. /opt/samurai/fuzzdb/wordlists-user-passwd/passwds/john.txt
   b. /opt/samurai/fuzzdb/wordlists-user-passwd/passwds/twitter.txt

    c. /opt/samurai/metasploit-framework/data/john/wordlists/password.lst

c. John can crack more efficiently if all the hashes (of the same type) are in the same file. For example, placing all the hashes that are the same type into file /tmp/hashes then letting john work on the whole file is more efficient. Gedit is installed on Samurai as is nano.
    a. ./john /tmp/hashes

d. If the hash format is not specified john will guess. Determining the likely format prior then specifying the format can help john know what type of hash to try. Otherwise John may guess the type incorrectly and no hashes will be cracked.
    a. ./john --format=raw-md5 /tmp/hashes

e. JTR will use its default wordlist (password.lst) by default but the user can specify other wordlists using the --wordlist switch
    a. ./john --format=raw-md5 --wordlist=/opt/samurai/fuzzdb/wordlists-user-passwd/passwds/twitter.txt /tmp/hashes

f. Use john --show to show cracked passwords. Tell John the format. John will output the cracked passwords for the hash format and file specified
    a. ./john --show --format=raw-md5 /tmp/hashes

g. John allows the user to mangle passwords using “Rules”. These can be specified in john.conf or in a separate file that is linked in john.conf. An example rule is seemed below. This rule adds one digit to each word in the dictionary. User defined rules are usually added just before the beginning of the first predefined rule.

    # Rule added by JD
    [List.Rules:AddOneDigit]
    Az"[0-9]"

h. Rules can be tested by sending the managed words to standard output

    a. root@samuraiwtf:/opt/john/john-1.8.0-jumbo-1/run# ./john --wordlist=/opt/samurai/fuzzdb/wordlists-user-passwd/passwds/twitter.txt --rule=AddOneDigit --stdout

i. Rules can be applied to the default wordlist or to an existing wordlist

    a. root@samuraiwtf:/opt/john/john-1.8.0-jumbo-1/run# ./john --format= raw-md5--wordlist=/opt/samurai/fuzzdb/wordlists-user-passwd/passwds/twitter.txt --rule=AddOneDigit /tmp/hashes