

Rainbow Tables & RainbowCrack Introduction

Rainbow tables reduce the difficulty in brute force cracking a single password by creating a large pre-generated data set of hashes from nearly every possible password. Rainbow Tables and RainbowCrack come from the work and subsequent paper by Philippe Oechslin.¹ The method, known as the Faster Time-Memory Trade-Off Technique, is based on research by Martin Hellman & Ronald Rivest done in the early 1980's on the performance trade-offs between processing time and the memory needed for cryptanalysis. In his paper published in 2003, Oechslin refined the techniques and showed that the attack could reduce the time to attack 99.9% of Microsoft's LAN Manager passwords (alpha characters only) to 13.6 seconds from 101 seconds. Further algorithm refinements also reduced the number of false positives produced by the system.

The main benefit of Rainbow Tables is that while the actual creation of the rainbow tables takes **much** more time than cracking a single hash, after they are generated you can use the tables over and over again. Additionally, once you have generated the Rainbow Tables, RainbowCrack is faster than brute force attacks and needs less memory than full dictionary attacks.

Rainbow Tables are popular with a particularly weak password algorithm known as Microsoft LM hash. LM stands for LAN Manager, this password algorithm was used in earlier days of Windows and still lives on only for compatibility reasons. By default Windows XP or even Windows Server 2003 keeps the LM hash of your passwords in addition to a more secure hash (NTLM or NTLMv2). This allows for the benefit of backwards compatibility with older operating systems on your network but unfortunately makes the job of password cracking easier if you can obtain the LM hashes instead of the NTLM hashes.

Microsoft's LAN Manager algorithm and its weaknesses

So why is the LM algorithm weak? "The LANManger scheme has several weaknesses, including converting all characters to uppercase, splitting passwords into 7-byte chunks, and not using an additional random element known as 'salt."²

These three issues give rainbow tables their cracking power. By converting all characters to uppercase you effectively cut your key space in half. So if you had passwords of only characters (A-Z, a-z) you would think you would have 52 possibilities, but in reality with LM, you only have 26 because password are converted to all uppercase. So my way secure password of PaSsWoRd would be converted automatically to PASSWORD.

Passwords longer than 7 characters are split into 2 chunks so a 14 character password is effectively turned into two, seven character passwords (and converted to uppercase). The chunks can also be attacked separately as you will see when we start cracking passwords.

Lastly, by not salting any of the passwords no extra complexity is added to stored passwords.

For some more background info check out the LM section of Wikipedia.org: http://en.wikipedia.org/wiki/LM hash

From the Rainbow Tables wiki:

"Rainbow tables use a refined algorithm by using a number of different reduction functions to create multiple parallel chains within a single "rainbow" table, reducing the probability of false positives from accidental chain collisions, and thus increasing the probability of a correct password crack. As well as increasing the probability of a correct crack for a given table size, the use of multiple reduction functions also greatly increases the speed of lookups.

¹ <u>http://lasecwww.epfl.ch/php_code/publications/search.php?ref=Oech03</u>

² <u>http://securitynews.weburb.org/show.php3?item=newsboard&p%5BmessageId%5D=3090</u>



Rainbow tables are specific to the hash function they were created for e.g., MD5 tables can crack only MD5 hashes. The theory of this technique was first pioneered by Philippe Oechslin³ as a fast form of time-memory tradeoff⁴, which he implemented in the Windows password cracker Ophcrack. The more powerful RainbowCrack program was later developed that can generate and use rainbow tables for a variety of character sets and hashing algorithms, including LM hash, MD5, SHA1, and NTLM."⁵

Let's check out some sample rainbow table configurations and see how they fare, as we go thru the tutorial all of this should make more sense.

LM configuration #0

charset	[ABCDEFGHIJKLMNOPQRSTUVWXYZ]
keyspace	8353082582
table size	610 MB
success probability	0.9990

Has a success probability of 99.90% and only takes up 610 MB.

LM configuration #1

charset	[ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789]
keyspace	80603140212
table size	3 GB
success probability	0.9904

Has a success probability of 99.04% and takes up 3 GB.

LM configuration #5

charset	[ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#\$%^&*()+=]
keyspace	915358891407 (2^39.7)
table size	24 GB
success probability	0.99909

Has a success probability of 99.1% and takes up 24 GB. This is starting to get large but 1) not THAT large with as cheap as hard drive space is and 2) with the character set involved. Don't forget this will work on passwords up to 14 characters as well. What starts to make a difference is how long it takes to compute these tables.

LM configuration #6

charset	[ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#\$%^&*()+=~`[]{}\\:;'''<>,.?/]
keyspace	7555858447479 (2^42.8)
table size	64 GB
success probability	0.999

³ <u>http://lasecwww.epfl.ch/philippe.shtml</u>

⁴ <u>http://lasecwww.epfl.ch/~oechslin/publications/crypto03.pdf</u>

⁵ <u>http://en.wikipedia.org/wiki/Rainbow_table</u>



Configuration Tables from http://www.antsight.com/zsl/rainbowcrack/

Has a success probability of 99.9% and takes up 64 GB. This character set includes all possible characters on a standard keyboard (not including alt+xxx characters). So this table set is likely to crack any windows password up 14 characters in minutes. This is great but on one computer it will take about 2 years to generate these tables (Faster Time-Memory Trade-Off Technique).

You can see demos of some of these configurations in action at the Project RainbowCrack website⁶

Using Rainbow Tables & RainbowCrack

Example 1:

First download RainbowCrack for your platform from <u>www.antsight.com/zsl/rainbowcrack/</u>.

We will use our LM alpha (configuration 0) rainbow tables.

** You will need to	b either create them or unzip ⁷ them and they the will look something like:
128,000,000 bytes	lm_alpha#1-7_0_2100x8000000_all.rt
128,000,000 bytes	lm_alpha#1-7_1_2100x8000000_all.rt
128,000,000 bytes	lm_alpha#1-7_2_2100x8000000_all.rt
128,000,000 bytes	lm_alpha#1-7_3_2100x8000000_all.rt
128,000,000 bytes	lm_alpha#1-7_4_2100x8000000_all.rt

If everything goes well, backup all files (recommended especially if you just made them and didn't download them) and then get ready to sort them.

To speed up the search of our rainbow table, we should sort the rainbow table with "rtsort.exe" in advance. In fact "rcrack.exe" only accepts sorted rainbow tables.

We sort the rainbow tables by using the following command:

Use these commands: rtsort lm_alpha#1-7_0_2100x8000000_all.rt rtsort lm_alpha#1-7_1_2100x8000000_all.rt rtsort lm_alpha#1-7_2_2100x8000000_all.rt rtsort lm_alpha#1-7_3_2100x8000000_all.rt rtsort lm_alpha#1-7_4_2100x8000000_all.rt

Each command will take several minutes to complete. The "rtsort.exe" utility will sort the file and write back to the original file.

Notice: If free memory size is smaller than the file size, we can't load the file into memory at a time. In which case extra free disk space as large as the file to be sorted is required to apply an external sort.

Once rtsort has completed you are ready to use rcrack against some hashes.

To see available options just type "rcrack"

⁶ Project RainbowCrack website <u>www.antsight.com/zsl/rainbowcrack/</u>

⁷ Free Rainbow tables for download via torrent <u>http://rainbowtables.shmoo.com/</u>



```
by Zhu Shuanglei <shuanglei@hotmail.com>
http://www.antsight.com/zsl/rainbowcrack/
usage: rcrack rainbow_table_pathname -h hash
      rcrack rainbow_table_pathname -1 hash_list_file
      rcrack rainbow_table_pathname -f pwdump_file
rainbow_table_pathname: pathname of the rainbow table(s), wildchar(*, ?) supported
                      use raw hash as input
-h hash:
-l hash_list_file:
                      use hash list file as input, each hash in a line
-f pwdump_file:
                      use pwdump file as input, this will handle LAN Manager ha
sh only
example: rcrack *.rt -h 5d41402abc4b2a76b9719d911017c592
         rcrack *.rt -l hash.txt
        rcrack *.rt -f hash.txt
```

Launch the program by issuing the command:

rcrack c:\rainbowcrack*.rt -l hashlist.txt

You should replace "c:\rainbowcrack\" with where you placed your sorted rainbow tables.

To crack some hashed windows passwords, the syntax is similar:

```
rcrack c:\rainbowcrack\*.rt -f pwdumpfile.txt
rcrack c:\rainbowcrack\*.rt -l justhashlist.txt
rcrack c:\rainbowcrack\*.rt -h 213D466DB5B288F0F82E44EC0938F4F4
```

Where pwdumpfile.txt is the results of using a hash dumping utility like pwdump2, pwdump3, samdump, etc to dump the LAN Manager s passwords.

If your password consists of only letters only, rcrack should be able to crack it with a success rate of 99.9%.

Let's try it against the following hash file in pwdump format (so use the –f option):

```
testuser1:"":0F20048EFC645D0A179B4D5D6690BDF3:1120ACB74670C7DD46F1D3F5038A5CE8:::
remote:"":E52CAC67419A9A224A3B108F3FA6CB6D:8846F7EAEE8FB117AD06BDD830B7586C:::
joeuser:"":E52CAC67419A9A224A3B108F3FA6CB6D:8846F7EAEE8FB117AD06BDD830B7586C:::
averageguy: " ": 299CCF964D9A359BAAD3B435B51404EE: A5C07214487C87B584E8877DE72DCA0B: ::
harderpass: "": B75838F7A57EE67993E28745B8BF4BA6: EC50F8A8149C93EF45AECB8AF96658E6: ::
demouser:"":261A6631FE44BA4993E28745B8BF4BA6:371D5760453C1B000BCC016F8E23A83C:::
randy: " ":98B5AFEB67293D6AAAD3B435B51404EE:A9F34664151F6360757B31644F37E025:::
Asmith: "": E165F0192EF85EBBAAD3B435B51404EE: E4EBE0E7EF708DC9FD240135D3D43D89:::
Bsmith: " ":136A8418CF76C4F7AAD3B435B51404EE:3431E75AD08DCA56EB53AEAAB9926589:::
csmith: "":BB26C063532826AA531C3383FDDBFF2A:A2746ED4129985C0251D2B968C4889FE:::
Dsmith: " ": A8EED815A197BD87AAD3B435B51404EE: F09A31889C35B8C9746B8F31FC3A868F: ::
Esmith: "": 5A9DB9F8BB5DF0CBAAD3B435B51404EE: 5FCC20A69EC76AD91214102B4D7DE24E:::
Fsmith: "": 213D466DB5B288F0F82E44EC0938F4F4: FAF10460760FA3F1ED804C7C724CB3D4:::
Gsmith: " ": 385A83A746BFA8F2AAD3B435B51404EE: 1CC1B3958B564125D307BA8D9D60DF69: ::
Hsmith: " :78BCCAEE08C90E29AAD3B435B51404EE:972E8E7D5568F70AC896B2C76E1395DC:::
Jsmith: ": 59E2DB85E9D49595B75E0C8D76954A50: 147D125645D463C33D72309525E9B0BC: ::
Ksmith:"":59E2DB85E9D49595B75E0C8D76954A50:147D125645D463C33D72309525E9B0BC:::
Lsmith: " ":13D855FC4841C7B1AAD3B435B51404EE:3DCEBC92C0ED8F52B1D759DD35CF3F0F:::
Msmith:"":D71808BF36F81510ADEE49688244F15A:45E8DA896575E2F5455B037FCC5AA51A:::
Nsmith: "":9C92FA4960AC2536AAD3B435B51404EE:C318744C4291EA46BC65082636CC9509:::
Osmith: " ":1153C3961EE58C3BAAD3B435B51404EE:672532E8C0C490BD47254DAED1CDCB36:::
```

Learn Security Online, Inc. ©



Psmith: " : 4A01C0E45FCA767AAAD3B435B51404EE: 39981702716E054CBE6840A3CFD60327::: Qsmith: " : 6842A19CC4C509E0AAD3B435B51404EE: 9FDA95FD6FCEE9C2C998CB8010F61F16::: Rsmith: " : BC472F3BF9A0A5F63832C92FC614B7D1: D2A80A79980CFA21CB58B7CB129E2CAD::: Ssmith: " : 09755C01D2789BD8AAD3B435B51404EE: 62F740C2EA31E10B54DB64CE12E867A6::: Tsmith: " : 13D855FC4841C7B1AAD3B435B51404EE: 3DCEBC92C0ED8F52B1D759DD35CF3F0F::: Usmith: " : 9E2204E2058AC9E9417EAF50CFAC29C3: 476541DEC5CB507A795FC1E989C9D36F::: Vsmith: " : 7F9CD2D7C93421D3F9DE51FBDAA2F725: 16FAABB24B95B82EFC50B074B7324517::: Wsmith: " : AC814111DF804A7482EFD6B2A69511D6: 15B194EB8D8F27761E32F76B001553A0::: Xsmith: " : AAD3B435B51404EEAAD3B435B51404EE: 2321504F2FA9437FBBA66EA1623407D3::: Ysmith: " : D5662E6B23655BF74EC0DA4207C2DE66: 75344B75B5A96614FE179C0188A9634A::: Zsmith: " : 9224FC255C58C50E42B35806901777F7: 0C105C9F4326C3AC100C2A5B7A04AD38::

The Answers so you can check your work.

		(0)			(0)
testuser1	testuserl	(2)	Ksmith	ABCdef123	(2)
remote	password	(2)	Lsmith	ABCdef	(1)
joeuser	password	(2)	Msmith	FOOTBALL!@#	(2)
averageguy	average	(1)	Nsmith	SOCCER	(1)
harderpass	rootwars	(2)	Osmith	CROKET	(1)
demouser	demopass	(2)	Psmith	COW123	(1)
randy	randy	(1)	Qsmith	HOWNOW	(1)
Asmith	ABCd	(1)	Rsmith	BROWNCOW	(2)
Bsmith	ef456	(1)	Ssmith	gHaNdI	(1)
csmith	ABC789!@#12	(2)	Tsmith	ABCdef	(1)
Dsmith	3!@#	(1)	Usmith	RTdotnet	(2)
Esmith	456!@#	(1)	Vsmith	!pa55word!	(2)
Fsmith	ABCdef!@#	(2)	Wsmith	EASYoneISNTit	(2)
Gsmith	дНдНдН	(1)	Xsmith	C@NTcR8ckm3CanU?	(X)no LM
Hsmith	ABC123	(1)	Ysmith	LSOISDABEST	(2)
Jsmith	ABCdef123^	(2)	Zsmith	RAINBOWTABLEZ	(2)

**32 users and 47 LM hashes 48 Total hashes. Xsmith will only be saved as NTLM because it's greater than 14 characters.

You should see something similar to the following:

:\Documents and Settings\NoOr -win>rcrack d:\torrents\lm_al			ACK-I.
m_alpha#1-7_0_2100x8000000_al		nes n a moowra	 I
28000000 bytes read, disk acc		s	
erifying the file			
earching for 41 hashes			
laintext of e52cac67419a9a22	is PASSWOR		
laintext of 4a3b108f3fa6cb6d			
laintext of b75838f7a57ee679			
laintext of 93e28745b8bf4ba6			
laintext of 261a6631fe44ba49			
laintext of 98b5afeb67293d6a			
laintext of e165f0192ef85ebb			
laintext of 385a83a746bfa8f2			
laintext of 13d855fc4841c7b1			
laintext of d71808bf36f81510 laintext of 9c92fa4960ac2536			
laintext of 1153c3961ee58c3b			
laintext of 6842a19cc4c509e0			
laintext of bc472f3bf9a0a5f6			
laintext of 09755c01d2789bd8			
	12 000001		

https://www.learnsecurityonline.com/



Figure 1.1: Rcrack at work with an lm_alpha rainbow table

		 _
tatistics		*
otal disk access time: otal cryptanalysis time: otal chain walk step: otal false alarm:	742.77 s 203410183	
emote password oeuser password werageguy average larderpass rootwars emouser demopass andy randy he smith ABCd hex ismith <notfound smith <notfound ismith <notfound ismith <notfound ismith <notfound< th=""><th>otfound> hex:54455354555345<notfound> hex:70617373776f7264 hex:70617373776f7264 hex:61766572616765 hex:61766572616765 hex:64656d6f70617373 x:72616e6479 :41424364 > hex:<notfound> Xnotfound> hex:<notfound> notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound< th=""><th></th></notfound<></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></th></notfound<></notfound </notfound </notfound </notfound 	otfound> hex:54455354555345 <notfound> hex:70617373776f7264 hex:70617373776f7264 hex:61766572616765 hex:61766572616765 hex:64656d6f70617373 x:72616e6479 :41424364 > hex:<notfound> Xnotfound> hex:<notfound> notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound> hex:<notfound< th=""><th></th></notfound<></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound></notfound>	

Figure 1.2: The results of our cracking attempt. 26 of our 41 hashes found in about 12 minutes. Also notice that the hash for the password "password" is the same because there is no salting with the LAN Manager hashing algorithm.

statistics

```
plaintext found:
                        26 of 41 (63.41%)
total disk access time:
                         62.51 s
total cryptanalysis time: 742.77s
total chain walk step:
                         203410183
total false alarm:
                         195135
total chain walk step due to false alarm: 142852030
result
                                  _____
testuser1 TESTUSE<notfound> hex:54455354555345<notfound>
remote password hex:70617373776f7264
remote
             password hex:70617373776f7264
joeuser
             average hex:61766572616765
averageguy
harderpass
               rootwars hex:726f6f7477617273
            demopass hex:64656d6f70617373
demouser
               randy hex:72616e6479
randy
               ABCd hex:41424364
Asmith
               <notfound> hex:<notfound>
Bsmith
csmith
               <notfound><notfound> hex:<notfound><notfound>
               <notfound> hex:<notfound>
Dsmith
               <notfound> hex:<notfound>
Esmith
Fsmith
               <notfound><notfound> hex:<notfound><notfound>
            qHqHqH hex:674867486748
Gsmith
Hsmith
               <notfound> hex:<notfound>
Jsmith
               <notfound><notfound> hex:<notfound><notfound>
Ksmith
               <notfound><notfound> hex:<notfound><notfound>
Lsmith
               ABCdef hex:414243646566
Msmith
               FOOTBAL<notfound> hex:464f4f5442414c<notfound>
Nsmith
               SOCCER hex: 534f43434552
```



Learn Security Online

Osmith	CROKET hex:43524f4b4554
Psmith	<notfound> hex:<notfound></notfound></notfound>
Qsmith	HOWNOW hex:484f574e4f57
Rsmith	BROWNCOW hex:42524f574e434f57
Ssmith	gHaNdI hex:6748614e6449
Tsmith	ABCdef hex:414243646566
Usmith	RTdotnet hex:5254646f746e6574
Vsmith	<notfound><notfound> hex:<notfound><notfound></notfound></notfound></notfound></notfound>
Wsmith	EASYoneISNTit hex:454153596f6e6549534e546974
Xsmith	hex:
Ysmith	LSOISDABEST hex:4c534f4953444142455354
Zsmith	RAINBOWTABLEZ hex:5241494e424f575441424c455a

Example 2:

We are going to build our own tables using Configuration #1

**Note if you built your configuration #0 tables using rtgen use winrtgen (see exercise 4)

configuration #1	
hash algorithm	lm
charset	alpha-numeric(ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789)
plaintext length range	1 - 7
key space	36^1 + 36^2 + 36^3 + 36^4 + 36^5 + 36^6 + 36^7 = 80603140212
t	2400
m	4000000
1	5
disk usage	m * 16 * 1 = 320000000 B = 3 GB
success rate	0.9904
mean cryptanalysis time	7.6276 s
mean cryptanalysis time on a low memory system (free memory size much smaller than 610MB)	13.3075 s
max cryptanalysis time	40.6780 s

Table pre-computation commands:

rtgen lm alpha-numeric 1 7 0 2400 40000000 all rtgen lm alpha-numeric 1 7 1 2400 40000000 all rtgen lm alpha-numeric 1 7 2 2400 40000000 all rtgen lm alpha-numeric 1 7 3 2400 40000000 all rtgen lm alpha-numeric 1 7 4 2400 40000000 all

On a 666 Mhz machine the table pre-computation time is about 15 days 17 hours, my P4 3.2 GHz with 1GB of RAM I created a table a day; so about 5 days.

Learn Security Online, Inc. ©		https://www.learnsecurityonline
	Learn Security	y <mark>Online</mark>
C:\WINDOWS\system32\cmd.exe	- rtgen Im alpha-numeric 1 7	1 2400 4000000 all
:\rainbowcrack-1.2-win\rain 1000000 all ash routine: lm	bowcrack-1.2-win>rtgen [lm alpha-numeric 1 7 1 2400 4 📥
ash length: 8 Ilain charset: ABCDEFGHIJKLM Ilain charset in hex: 41 42 55 56 57 58 59 5a 30 31 32	43 44 45 46 47 48 49 4a	4b 4c 4d 4e 4f 50 51 52 53 5
hash length: 8 plain charset: ABCDEFGHIJKLM plain charset in hex: 41 42 4 55 56 57 58 59 5a 30 31 32 plain length range: 1 - 7 plain charset name: alpha-nu plain space total: 806031402	43 44 45 46 47 48 49 4a 33 34 35 36 37 38 39 meric	4b 4c 4d 4e 4f 50 51 52 53 5
hash length: 8 plain charset: ABCDEFGHIJKLM plain charset in hex: 41 42 4 55 56 57 58 59 5a 30 31 32 plain length range: 1 - 7 plain charset name: alpha-nu plain space total: 806031402 rainbow table index: 1 reduce offset: 65536	43 44 45 46 47 48 49 4a 33 34 35 36 37 38 39 meric	4b 4c 4d 4e 4f 50 51 52 53 5
hash length: 8 plain charset: ABCDEFGHIJKLM plain charset in hex: 41 42 4 55 56 57 58 59 5a 30 31 32 plain length range: 1 - 7 plain charset name: alpha-nu plain space total: 806031402: rainbow table index: 1 reduce offset: 65536 generating 100000 of 40000000 rainbow c	43 44 45 46 47 48 49 4a 33 34 35 36 37 38 39 meric 12 hains generated <3 m 4 s	
hash length: 8 plain charset: ABCDEFGHIJKLM plain charset in hex: 41 42 4 55 56 57 58 59 5a 30 31 32 plain length range: 1 - 7 plain charset name: alpha-nu plain space total: 806031402 rainbow table index: 1 reduce offset: 65536 generating 200000 of 40000000 rainbow c 200000 of 40000000 rainbow c	43 44 45 46 47 48 49 4a 33 34 35 36 37 38 39 meric 12 hains generated (3 m 4 s hains generated (3 m 3 s hains generated (3 m 3 s	
hash length: 8 plain charset: ABCDEFGHIJKLM plain charset in hex: 41 42 4 55 56 57 58 59 5a 30 31 32 plain length range: 1 - 7 plain charset name: alpha-nu plain space total: 806031402 rainbow table index: 1 reduce offset: 65536 generating 100000 of 40000000 rainbow c 200000 of 40000000 rainbow c 4000000 of 40000000 rainbow c	43 44 45 46 47 48 49 4a 33 34 35 36 37 38 39 meric 12 hains generated (3 m 4 s hains generated (3 m 3 s hains generated (3 m 3 s hains generated (3 m 3 s	
hash length: 8 plain charset: ABCDEFGHIJKLM plain charset in hex: 41 42 4 55 56 57 58 59 5a 30 31 32 plain length range: 1 - 7 plain charset name: alpha-nu plain space total: 806031402 rainbow table index: 1 reduce offset: 65536 generating 200000 of 40000000 rainbow c 300000 of 40000000 rainbow c 500000 of 40000000 rainbow c 500000 of 40000000 rainbow c	43 44 45 46 47 48 49 4a 33 34 35 36 37 38 39 meric 12 hains generated (3 m 4 s hains generated (3 m 3 s	
hash length: 8 plain charset: ABCDEFGHIJKLM plain charset in hex: 41 42 4 55 56 57 58 59 5a 30 31 32 plain length range: 1 - 7 plain charset name: alpha-nu plain space total: 806031402 rainbow table index: 1 reduce offset: 65536 generating 200000 of 40000000 rainbow c 300000 of 40000000 rainbow c 500000 of 40000000 rainbow c 500000 of 40000000 rainbow c	43 44 45 46 47 48 49 4a 33 34 35 36 37 38 39 meric 12 hains generated (3 m 4 s hains generated (3 m 3 s	

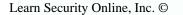
Figure 2.1: Creating our LM alpha-numeric rainbow tables.

Now run that table against the same hash file, don't forget to sort them first. You should crack most, if not all, of the alpha-numeric passwords, as opposed to alpha passwords only from configuration #0.

🔍 Command Pro	ompt (2)	- 🗆 🗙
statistics		-
total cryptan total chain w total false a	nd: 32 of 41 (78.05%) cess time: 233.84 s alysis time: 233.05 s alk step: 211003249 larm: 104620 alk step due to false alarm: 91734872	
testuser1 remote joeuser averageguy harderpass demouser randy Asmith Bsmith Bsmith Dsmith Esmith Gsmith Hsmith	password hex:70617373776f7264 password hex:70617373776f7264	•

Figure 2.2: The results of our attempts. 32 of 41 passwords were found. Note that I ran this on my 3.2 GHz machine because I created the tables on it and didn't want copy 3GB of rainbow tables to the slow computer.

statistics
plaintext found: 32 of 41 (78.05%)
total disk access time: 233.84 s
total cryptanalysis time: 233.05 s
total chain walk step: 211003249
total false alarm: 104620





https://www.learnsecurityonline.com/

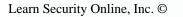
total chain walk step due to false alarm: 91734872

result	
testuser1	testuser1 hex:746573747573657231
remote	password hex:70617373776f7264
joeuser	password hex:70617373776f7264
averageguy	average
harderpass	rootwars hex:726f6f7477617273
demouser	demopass hex:64656d6f70617373
randy	randy hex:72616e6479
Asmith	ABCd hex:41424364
Bsmith	ef456 hex:6566343536
csmith	<notfound><notfound> hex:<notfound><notfound></notfound></notfound></notfound></notfound>
Dsmith	<notfound> hex:<notfound></notfound></notfound>
Esmith	<notfound> hex:<notfound></notfound></notfound>
Fsmith	<notfound><notfound> hex:<notfound><notfound></notfound></notfound></notfound></notfound>
Gsmith	дНдНдН hex:674867486748
Hsmith	ABC123 hex:414243313233
Jsmith	ABCdef123 hex:414243646566313233
Ksmith	ABCdef123 hex:414243646566313233
Lsmith	ABCdef hex:414243646566
Msmith	FOOTBAL <notfound> hex:464f4f5442414c<notfound></notfound></notfound>
Nsmith	SOCCER hex:534f43434552
Osmith	CROKET hex:43524f4b4554
Psmith	COW123 hex:434f57313233
Qsmith	HOWNOW hex:484f574e4f57
Rsmith	BROWNCOW hex:42524f574e434f57
Ssmith	gHaNdI hex:6748614e6449
Tsmith	ABCdef hex:414243646566
Usmith	RTdotnet hex:5254646f746e6574
Vsmith	<notfound><notfound> hex:<notfound><notfound></notfound></notfound></notfound></notfound>
Wsmith	EASYoneISNTit hex:454153596f6e6549534e546974
Xsmith	hex:
Ysmith	LSOISDABEST hex:4c534f4953444142455354
Zsmith	RAINBOWTABLEZ hex:5241494e424f575441424c455a

Example 3:

Compare the results of the same hash file with Cain in Brute force mode, John the Ripper, and LC4.

Cain, in brute-force mode with an alpha-numeric character set, says it will take about 10 hours.





Learn Security Online

Ξ aíŋ				
File Vi	ew Configure Tools Help			
🔄 🏟 🐼 B	rute-Force Attack		×	
Protected	Charset Charse		Password length Min 1 Max 7	Vireless
MTLMv2 I MS-Cach	C Custom		Start from	20ACB74670 346F7EAEE8F 346F7EAEE8F 5C07214487C
Cisco PIX Pipe APOP-MC CRAM-MI	Keyspace 80603140212	Current password		250F8A8149C 11D5760453C 9F34664151F FEBE0E7EF70.
OSPF-ME → RIPv2-MI → VRRP-HM	Key Rate 2070958 Pass/Sec	Time Left 10.8102 ho	purs	131E75AD08D 2746ED41299
	Plaintext of 93E28745B8BF4BA6 is 9 Plaintext of 417EAF50CFAC29C3 is Plaintext of 3832C92FC614B7D1 is Plaintext of 179B4D5D6690BDF3 is 1 Plaintext of B75E0C8D76954A50 is Plaintext of E165F0192EF85EBB is Plaintext of 4EC0DA4207C2DE66 is 1	Г Л R1 23 ABCD		99A31889C35 FCC20A69EC7 AF10460760F CC183958856 72E8E7D5568 I7D125645D4 I7D125645D4 DCEBC92C0E.
Radius Sl G IKE-PSK I MSSQL H			Stop Exit	E8DA896575 🚩
http://www.oxid.it	LM & NTLM Hashes			

Figure 3.1: Cain in brute-force mode.

🔹 📀 🕇 🥹	B ₆₄ 2 mm	📟 🅐 🚾 🏧 📾		°à 😵 🗶 🔏	20 ?	n.
rotected Storage	Network 🙀 Sniffer	📸 LSA Secrets 🥳	Crac	:ker 🙋 Tracerout	e 🔝 CCDU 🕅	Wireless
cker 📉	User Name	LM Password	< 8	NT Password	LM Hash	NT Hash
LM & NTLM Hash	S Hsmith	ABC123	*	ABC123	78BCCAEE08C9	972E8E7D5568
NTLMv2 Hashes (0)	🐴 Jsmith	ABCDEF123		ABCdef123	59E2DB85E9D4	147D125645D4
MS-Cache Hashes (Ksmith	ABCDEF123		ABCdef123	59E2DB85E9D4	147D125645D4
PWL files (0)	Lsmith	ABCDEF	*	ABCdef	13D855FC4841	3DCEBC92C0E.
Cisco IOS-MD5 Hasl	X Msmith	FOOTBAL??????			D71808BF36F8	45E8DA896575
Cisco PIX-MD5 Hash APOP-MD5 Hashes	Nsmith	SOCCER	*	SOCCER	9C92FA4960AC	C318744C4291
CRAM-MD5 Hashes	Sosmith	CROKET	*	CROKET	1153C3961EE5	672532E8C0C4
OSPF-MD5 Hashes	Psmith	COW123	*	COW123	4A01C0E45FCA	39981702716E.
RIPv2-MD5 Hashes	A Qsmith	HOWNOW	*	HOWNOW	6842A19CC4C5	9FDA95FD6FCE
VRRP-HMAC Hashe:	Rsmith	BROWNCOW		BROWNCOW	BC472F3BF9A0	D2A80A79980C
VNC-3DES (0)	Ssmith	GHANDI	*	gHaNdI	09755C01D278	62F740C2EA31
MD2 Hashes (0)	S Tsmith	ABCDEF	*	ABCdef	13D855FC4841	3DCEBC92C0E.
MD4 Hashes (0)	SUsmith	RTDOTNET		RTdotnet	9E2204E2058A	476541DEC5CB
MD5 Hashes (0)	X Vsmith				7F9CD2D7C934	16FAABB24B95
SHA-1 Hashes (0)	👫 Wsmith	EASYONEISNTIT		EASYoneISNTit	AC814111DF80	15B194EB8D8F
SHA-2 Hashes (0)	X Xsmith	* empty *	*		AAD3B435B514	2321504F2FA9
RIPEMD-160 Hashe	P Ysmith	LSOISDABEST		LSOISDABEST	D5662E6B2365	75344B75B5A9
Kerb5 PreAuth Hasł —	🔥 Zsmith	RAINBOWTABLEZ		RAINBOWTABLEZ	9224FC255C58	0C105C9F4326
Radius Shared-Key	V					

http://www.oxid.it Figure 3.2: After 9+ hours it cracked 27 of the 41 hashes



John the Ripper, in default mode, was able to quickly (about 3 minutes) crack 32 of the 48 hashes.

loaded 48 pa	sswords with no differ	ent salts (NT	LM DES [24/32 4K1)	
ANDY	(randy)	one sures (m		-
VERAGE	(averageguy)			
ESTUSE	(testuser1:1)			
1	(testuser1:2)			
BC123	(Hsmith)			
ASSWOR	(remote:1)			
PASSWOR	(joeuser:1)			
OCCER	(Nsmith)			
OOTBAL	(Msmith:1)			
BCDEF	(Lsmith)			
BCDEF	(Tsmith)			
RAINBOW	(Zsmith:1)			
IBCD	(Asmith)			
BCDEF1	(Jsmith:1)			
BCDEF1	(Ksmith:1)			
BCDEF!	(Fsmith:1)			
)	(remote:2)			
)	(joeuser:2)			
	(Üsmith:2)			
	(harderpass:2)			
	(demouser:2)			
	(Rsmith:2)			
	(Xsmith)			
EST	(Ysmith:2)			
3	(Jsmith:2)			
3	(Ksmith:2)			
ROWNCO	(Rsmith:1)			
#	(Fsmith:2)			
OWNOW	(Qsmith)			
ROKET	(Osmith)			
HANDI	(Ssmith)			
D!	(Usmith:2)	Contractor of the second second	5.0000.0000000000000000000000000000000	
uesses: 32	time: 0:00:02:38 (3)	c/s: 8803701	trying: BH9IRFU - SKO	
EF456	(Bsmith)			

Figure 3.3: JTR at work.

After 24 hours we had 45 of the 48 hashes.

EF456 (Bsmith) COW123 (Psmith) INTIT (Wsmith:2) IABLEZ (Zsmith:2) EASYONE (Wsmith:1) guesses: 37 time: 0:00:21:41 (3) c/s: 7499555 trying: EFLGRBA - EUCNU93 P#12 (csmith:2) 319# (Dsmith) GHGHGH (Gsmith) DEMOPAS (demouser:1) RODTWE (Usmith:1) guesses: 44 time: 0:19:45:10 (3) c/s: 3851222 trying: AGWUH0A - N067I8Y guesses: 44 time: 0:19:45:10 (3) c/s: 3851222 trying: BNF9TZL - M6A50BH LSOISDA (Ysmith:1) guesses: 45 time: 0:23:13:08 (3) c/s: 23921977 trying: L20LY10 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 23921975 trying: 3ETI3ZP - 4K00-ZP guesses: 45 time: 0:23:19:56 (3) c/s: 2383601 trying: 3C7RIGG - 3BNBJ6E	Commanu	Prompt					- 0	×
ÈP456 (Bsmith) COW123 (Psmith) ISNTIT (Wsmith:2) TABLEZ (Zsmith:2) EASYONE (Wsmith:1) guesses: 37 time: 0:00:21:41 (3) c/s: 7499555 trying: EFLGRBA - EUCNU93 0#12 (csmith:2) (csmith:2) (3) c/s: 7499555 trying: EFLGRBA - EUCNU93 0#12 (csmith:2) (1) (1) (1) (1) (1) 1?01 (Csmith:2) (1) (1) (1) (1) (1) 1?01 (Csmith:2) (1) (1) (1) (1) (1) 0EMOPAS (demouser:1) (1) (1) (1) (1) (1) RIDOTNE (Usmith:1) (1) (1) (1) (1) (1) (1) guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH LSOISDA (Ysmith:1) (1) (2) c/s: 23972777 trying: L20LY10 - JXXHFZK guesses: 45 time: 0:23:13:08 (3) c/s: 2392985 trying: 3CTRIGC - 3BNBJGE gues	D!	(Usmith:2)						
COW123 (Psmith) ISNTIT (Wsmith:2) TABLEZ (Zsmith:2) EASYONE (Wsmith:1) guesses: 37 time: 0:00:21:41 (3) c/s: 7499555 trying: EFLGRBA - EVCNU93 Q#12 (csmith:2) 12 (csmith:2) 14 (Msnith:2) 310# (Dsmith) DEMOPAS (demouser:1) RODTMAR (harderpass:1) RIDOTNE (Usmith:1) guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 SOISDA (Ysmith:1) guesses: 45 time: 0:20:03:11 (3) c/s: 2734211 guesses: 45 time: 0:23:13:08 (3) c/s: 2392985 trying: GE88ZA - FDWXTUF guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ETI3ZF - 4K00-ZF guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7RIGG - 3BNBJ6E		time: 0:00:02:38	(3) c/:	: 8803701	trying:	BH9IRFU - SKONOIN		
SNTIT (Wsmith:2) ABLEZ ABLEZ (Zsmith:2) (Wsmith:1) uesses: 37 time: 0:00:21:41 (3) c/s: 7499555 trying: EFLGRBA - EUCNU93 #12 (csmith:2) #14 (Dsmith) HCH (Gsmith) HGH (Gsmith) HGH (Gsmith) HGH (Gsmith) Uesses: 44 time: 0:05:17:11 Uesses: 44 time: 0:05:17:11 Uesses: 44 time: 0:05:17:11 Uesses: 44 time: 0:19:45:10 Vesses: 44 time: 0:20:03:11 Vesses: 45 time: 0:20:03:11 Uesses: 45 time: 0:23:13:08 Uesses: 45								
ABLEZ (Zsmith:2) GASYONE (Wsmith:1) Gasyone (Wsmith:1) guesses: 37 time: 0:00:21:41 (3) c/s: 7499555 trying: EFLGRBA - EUCNU93 WH12 (csmith:2) YP# (Msmith:2) YP# (Msmith:2) YP# (Msmith:2) YP# (Msmith:2) YP# (Dsmith) SHGHGH (Gsmith) DEMOPAS (demouser:1) QUESSES: 44 time: 0:05:17:11 (3) c/s: 3851222 trying: AGWUH0A - N067I8Y guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH SOISDA (Ysmith:1) c/s: 2734211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LY!0 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3CTRIGG - 3BNEJ6E	:0W123	(Psmith)						
GASYONE (Wsmith:1) guesses: 37 time: 0:00:21:41 (3) c/s: 7499555 trying: EFLGRBA - EUCNU93 912 (csmith:2) 914 (Msmith:2) 914 (Msmith:1)		(Wsmith:2)						
guesses: 37 time: 0:00:21:41 (3) c/s: 7499555 trying: EFLGRBA - EUCNU93 2412 (csmith:2) 2412 (Msmith:2) 310# (Dsmith) 310# (Dsmith:1) 310# (Usmith:1) 310# (S) 310# (S) <tr< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td></tr<>								1
#12 (csmith:2) .1°E# (Msmith:2) .1°E# (Dsmith) SHGHGH (Gsmith) SHGHA (Gsmith) SHOPAS (demouser:1) ROOTWAR (harderpass:1) RTDOTNE (Usmith:1) guesses: 44 time: 0:05:17:11 (3) c/s: 3851222 trying: AGWUH0A - N06718Y guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH LSOISDA (Ysmith:1) juesses: 45 time: 0:20:03:11 (3) c/s: 2374211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LY!0 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3CTRIGG - 3BNEJ6E								
19# (Msmith:2) 19# (Dsmith) 3HGHGH (Gsmith) DEMOPAS (demouser:1) NOOTWAR (harderpass:1) RTDOTNE (Usmith:1) Juesses: 44 time: 0:05:17:11 (3) c/s: 3851222 trying: AGWUH0A - N06718Y Juesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH SOISDA (Ysmith:1) c/s: 2734211 trying: GCE88ZA - FDWXTUF Juesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LV #0 - JXXHFZK Juesses: 45 time: 0:23:19:56 (3) c/s: 239285 trying: 3CTRIGG - 3BNEJ6E			(3) c/s	: 7499555	trying:	EFLGRBA - EVCNU93		
11 E# (Dsmith) SHGHGH (Gsmith) SHGHGH (Gsmith) DEMOPAS (demouser:1) RODTWAR (harderpass:1) (Usmith:1) (usmith:1) puesses: 44 time: 0:19:15:10 (3) c/s: 3851222 trying: AGWUH0A - N067I8Y puesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH .SOISDA (Ysmith:1) (Ysmith:1) (Ysmith:1) puesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LV #0 - JXXHFZK puesses: 45 time: 0:23:19:56 (3) c/s: 2383601 trying: 3CTRIGG - 3BNEJ6E								
GHGHGH (Gsmith) DEMOPAS (denouser:1) ROOTWAR (harderpass:1) RIDOTNE (Usnith:1) guesses: 44 time: 0:05:17:11 (3) c/s: 3851222 trying: AGWUH0A - N06718Y guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH GUESSES: 45 time: 0:20:03:11 (3) c/s: 2734211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2392985 trying: J20LY10 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2382601 trying: 3C7R1GG - 3BNBJ6E								
DEMOPAS (demouser:1) ROOTWAR (harderpass:1) RIDOTNE (Usmith:1) guesses: 44 time: 0:19:45:10 (3) c/s: 3851222 trying: AGWUH0A - N06718Y guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH LSOISDA (Ysmith:1) c/s: 2734211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LY!0 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3C7RIGG - 3BNBJ6E								
ROOTWAR (harderpass:1) (Usmith:1) guesses: 44 (harderpass:1) (Usmith:1) guesses: 44 (harderpass:1) (Usmith:1) (Usmith:1) guesses: 45 (harderpass:1) (Usmith:1) guesses: 45 (harderpass:1) (Usmith:1) (Usmith:1) guesses: 45 (harderpass:1) (Usmith:1) (Us								
NIDOTNE (Usmith:1) guesses: 44 time: 0:05:17:11 (3) c/s: 3851222 trying: AGWUH0A - N06718Y guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH LSOISDA (Ysmith:1) guesses: 45 time: 0:20:03:11 (3) c/s: 2734211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LY10 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ETI3ZP - 4K00-ZP guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7R1GG - 3BNBJ6E								
guesses: 44 time: 0:05:17:11 (3) c/s: 3851222 trying: AGWUH0A - N06718Y guesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF91ZL - M6A50BH LSOISDA (Ysmith:1) guesses: 45 time: 0:20:03:11 (3) c/s: 2734211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LY!0 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ETI3ZP - 4K00-ZP guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7R1GG - 3BNBJ6E			1)					
juesses: 44 time: 0:19:45:10 (3) c/s: 2740978 trying: BNF9TZL - M6A50BH (Ysmith:1) juesses: 45 time: 0:20:03:11 (3) c/s: 2734211 trying: GCE88ZA - FDWXTUF juesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LV+0 - JXXHFZK juesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ET13ZP - 4K00-ZP juesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3CTRIGC - 3BNBJ6E								
LSOISDA (Ysmith:1) guesses: 45 time: 0:20:03:11 (3) c/s: 2734211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LY!0 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ETI3ZP - 4K00-ZP guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7R1GG - 3BNBJ6E					trying:			
guesses: 45 time: 0:20:03:11 (3) c/s: 2734211 trying: GCE88ZA - FDWXTUF guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LY!0 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ETI3ZP - 4K00-ZP guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7R1GG - 3BNBJ6E			(3) c/:	: 2740978	trying:	BNF9TZL - M6A5OBH		
guesses: 45 time: 0:23:13:08 (3) c/s: 2397177 trying: L20LV40 - JXXHFZK guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ET13ZP - 4K00-ZP guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7R1GG - 3BNBJ6E								
guesses: 45 time: 0:23:19:56 (3) c/s: 2392985 trying: 3ETI3ZP - 4K00-ZP guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7R1GG - 3BNBJ6E								
guesses: 45 time: 0:23:36:10 (3) c/s: 2383601 trying: 3C7R1GG - 3BNBJ6E								
guesses: 45 time: 1:00:00:18 (3) c/s: 2371816 trying: M2XSCA - CHITRPR								
uesses: 45 time: 1:00:00:21 (3) c/s: 2371770 trying: M62NAPI - C1ZNTON			(3) c/:	: 2371770	trying:	M62NAPI - CIZNIUN		
Session aborted	ession abo	rted						

Figure 3.4: JTR after 24 hours of cracking



The results of our efforts! All but 3 of the hashes were cracked in 24 hours by John. The "Xsmith" account with 15 characters was not cracked. With enough time we should have been able to find the passwords for "csmith" and "Vsmith"

Note that this really wasn't a fair assessment since john will try characters not in our rainbow tables. If you want a really fair assessment, you should modify john's ini file. But I don't plan on doing it that. The point of the tables is the speed. But honestly, for this password file, John did really well.

Command Prompt	- 🗆 🗙
:\Documents and Settings\NoOne\Desktop\CLI\john-16\run>john -show hash	es4rainbo
lab.txt	
estuser1:TESTUSER1:"":1120ACB74670C7DD46F1D3F5038A5CE8:::	
emote:PASSWORD:"":8846F7EAEE8FB117AD06BDD830B7586C:::	
oeuser:PASSWORD:""'8846F7EAEE8FB117AD06BDD830B7586C:::	
verageguy:AVERAGE:"":A5C07214487C87B584E8877DE72DCA0B:::	
arderpass:ROOTWARS:'''':EC50F8A8149C93EF45AECB8AF96658E6:::	
emouser:DEMOPASS:"":371D5760453C1B000BCC016F8E23A83C:::	
andy:RANDY:'''':A9F34664151F6360757B31644F37E025:::	
smith:ABCD:"":E4EBE0E7EF708DC9FD240135D3D43D89:::	
smith:EF456:"":3431E75AD08DCA56EB53AEAAB9926589:::	
smith:??????@#12:"":A2746ED4129985C0251D2B968C4889FE:::	
smith:3!@#:""':F09A31889C35B8C9746B8F31FC3A868F:::	
smith:ABCDEF!@#:''':FAF10460760FA3F1ED804C7C724CB3D4:::	
smith:GHGHGH:'''':1CC1B3958B564125D307BA8D9D60DF69:::	
smith:ABC123:"":972E8E7D5568F70AC896B2C76E1395DC:::	
smith:ABCDEF123:"":147D125645D463C33D72309525E9B0BC:::	
smith:ABCDEF123:"":147D125645D463C33D72309525E9B0BC:::	
smith:ABCDEF:"":3DCEBC92C0ED8F52B1D759DD35CF3F0F:::	
smith:FOOTBALL!@#:"":45E8DA896575E2F5455B037FCC5AA51A::::	
smith:SOCCER:'''':C318744C4291EA46BC65082636CC9509:::	
smith:CROKET:"":672532E8C0C490BD47254DAED1CDCB36:::	
smith:COW123:"":39981702716E054CBE6840A3CFD60327:::	
smith:HOWNOW:"":9FDA95FD6FCEE9C2C998CB8010F61F16:::	
smith:BROWNCOW:"":D2A80A79980CFA21CB58B7CB129E2CAD:::	
smith:GHANDI:"":62F740C2EA31E10B54DB64CE12E867A6:::	
smith:ABCDEF:"":3DCEBC92C0ED8F52B1D759DD35CF3F0F::: smith:BTD0TNET:"":476541DEC5CB507A795FC1E989C9D36F:::	
smith:???????RD!:"":16FAABB24B95B82EFC50B074B7324517:::	
smith:EASYONEISNTIT:"":15B194EB8D8F27761E32F76B001553A0:::	
smith::"":2321504F2FA9437FBBA66EA1623407D3:::	
smith:LSOISDABEST:"":75344B75B5A96614FE179C0188A9634A::::	
smith:RAINBOWTABLEZ:"":0C105C9F4326C3AC100C2A5B7A04AD38:::	
SMICH-ANTADOWINDIEZ- OGIØJC/14JZ0CJAGIØØCZAJD/AØMAJ30···	
5 passwords cracked, 3 left	

Figure 3.5: the results 45 passwords cracked

Let's see how LC4 fairs against our password file, I did turn off the dictionary and hybrid modes on LC4 and selected alphanumeric as our characters in the session options, so this should be a pretty fair "time to crack the same hashes" test.



Learn Security Online

	nport Session Help	🖡 🕨 II 😽 🔯			
main	User Name	LM Password	<8 NTLM Password	Audit Time	DICTIONARY STATUS
	testuser1				words total
	remote				0 words_done
	joeuser				
	averageguy		×		<u>% done</u> 0.000%
	harderpass				
	demouser				BRUTE FORCE
	randy		×		
	Asmith		×		time_left
	Bsmith		×		
	csmith				%_done
	Dsmith		x		_current_test
	Esmith		×		to success a
	Fsmith				kevrate
	Gsmith		x		
	Hsmith		x		SUMMARY
	Jsmith				total_users
	Ksmith				audited users
	Lsmith		x		0 % done
	Msmith				0.000%
	Nsmith		×		
	Osmith		x		User Info Check Dictionary
	Psmith		x		Hybrid
	Qsmith		x	~	Brute Force

Figure 3.6: Loading the password file of 32 users into LC4.

	ort Session Help					
🖲 🗲 🍋	🖬 📲 📩 🗖 🖂 🖡	l 🕨 🖬 📢 👺 🖁) 😰 🗳		
omain	User Name	LM Password	<8	NTLM Password	Audit Time	DICTIONARY STATUS
3	testuser1	TESTUSER1		testuser1	Od Oh Om Os	words_total
1	remote	777777D				C words_done
3	joeuser	777777D				
7	averageguy	AVERAGE	×			<u>%_done</u> 0.000%
1	harderpass	7777775				0.000%
1	demouser	7777775				BRUTE FORCE
2	randy	RANDY	×	randy	Od Oh Om Os	_ <u>time_elapsed</u> Od Oh 7m43s
2	Asmith	ABCD	x	ABCd	Od Oh Om Os	time left
3	Bsmith	EF456	×	ef456	0d 0h 0m 23s	Od11h Om 1s
3	csmith					<u>%_done</u> 1.1556%
3	Dsmith		×			current test
3	Esmith		×			MDKNXY
3	Fsmith					<u>kevrate</u> 2435291 k/s
3	Gsmith	GHGHGH	×	gHgHgH	0d 0h 4m 58s	01111110.004
3	Hsmith		x			SUMMARY
3	Jsmith	77777723				total users
3	Ksmith	77777723				audited users
3	Lsmith	ABCDEF	×	ABCdef	0d 0h 6m 11s	10 2. done
3	Msmith					31.250%
3	Nsmith	SOCCER	×	SOCCER	0d 0h 1m 48s	and the second second
3	Osmith	CROKET	×	CROKET	0d 0h 0m 53s	User Info Check
3	Psmith		×			Hybrid
3	Qsmith		×			Brute Force
						Hybrid Brute Force

Figure 3.7: LC4 estimated about 11 hours to bruteforce crack the passwords using an alphanumeric character set.

In 11 hours we were able to crack 26 out of the 32 user accounts but the Xsmith account was not cracked because we did not attempt an NTLM attack.



Learn Security Online

	Session Help		16 16			
	🗐 🗼 🗖 🛛 🖡 User Name	LM Password	<8	NTLM Password	Audit Time	
	testuser1	TESTUSER1	1.0	testuser1	Od Oh Om Os	-
<u>.</u>		PASSWORD			0d 1h 4m 52s	words_total
	remote			password		words_done
1	joeuser	PASSWORD		password	0d 1h 4m 52s	0 %_done
<u>.</u>	averageguy	AVERAGE	×	5222222222	and a second second	0.000%
-	harderpass	ROOTWARS		rootwars	0d 1h 5m 18s	BRUTE FORCE
- · · ·	demouser	DEMOPASS		demopass	0d 2h 9m 47s	1
	randy	RANDY	х	randy	Od Oh Om Os	<u>_time_elapsed</u> Od10h22m23s
<u> </u>	Asmith	ABCD	×	ABCd	Od Oh Om Os	time_left
	Bsmith	EF456	×	ef456	Od Oh Om 24s	Od Oh Om Os % done
	csmith					100.0000%
	Dsmith		×			<u>current_test</u>
<u>,</u>	Esmith		×			keyrate
	Fsmith					
	Gsmith	GHGHGH	×	gHgHgH	0d 0h 4m 31s	SUMMARY
4	Hsmith	ABC123	×	ABC123	0d 0h 13m 39s	
	Jsmith	ABCDEF123		ABCdef123	0d 8h 47m 50s 📃	<u>total_users</u> 32
() () () () () () () () () ()	Ksmith	ABCDEF123		ABCdef123	0d 8h 47m 50s	audited_users
	Lsmith	ABCDEF	×	ABCdef	0d 0h 5m 37s	24
-	Msmith	FOOTBAL???????				<u>% done</u> 75.000%
	Nsmith	SOCCER	x	SOCCER	0d 0h 1m 41s	
<u>e</u>	Osmith	CROKET	x	CROKET	0d 0h 0m 50s	User Info Check Dictionary
9	Psmith	COW123	x	COW123	0d 0h 13m 39s	Hybrid
	Qsmith	HOWNOW	x	HOWNOW	0d 0h 8m 16s 🛛 🗸	Srute Force

Figure 3.8: So 11 hours versus 12.3795 minutes with the rainbow tables.

Example 4:

Using Cain and Abel's Winrtgen to create your Rainbow Tables. Winrtgen supports Rainbow Tables for the following hashing/encryption algorithms: LM, FastLM, NTLM, CiscoPIX, MD2, MD4, MD5, SHA-1, SHA-2 (256), SHA-2 (384), SHA-2 (512), MySQL (323), MySQL (SHA1) and RIPEMD160.

🖩 Winrtgen v1.8 (Rainbow Tables Ge	nerator) by mao	
Filename	Status	
Add Table Remove Remove All	About	OK Exit

Figure 4.1: Open Winrtgen and select Add Table

earn Security Online, Inc. ©	https://www.learnsecurityonli	ine.com
Learn	Security Online	
ainbow Table properties	X	
Hash Min Len Max Len Index Cha Im 1 7 0 240	Len Chain Count N° of tables	
Charset		
alpha-numeric	Edit	
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789		
Table properties Key space: 80603140212 keys Disk space: 610.35 MB Success probability: 0.607082 (60.71%)		
Benchmark		
Hash speed: 717360 hash/sec		
Step speed: 490484 step/sec Table precomputation time: 2.26534 days		
Total precomputation time: 2.26534 days		
Max cryptanalysis time: 5.87175 seconds		

Figure 4.2: Select LM, 1 to 7 for Min/Max Length, Chain Length 2400 and Chain Count 40,000,000.

As you see with one table we get about 60% success rate and it will take about 2 days to create the table on a P3 1GHz machine. Feel free to manipulate Chain Length (remember that it will increase success rate but increase computation time) to whatever you can handle for table pre-computation time. I will leave it at 2400 for now. But 60% isn't that great, for a 99.06% success rate you will need to create 5 tables (3 GB of space) and it will take about 12 days to create the tables. For a 99.63% success rate you will need to create 6 tables (3.57 GB of space) and it will take about 14 days to generate the tables. I will go with 5 tables for a success rate of 99.06%.

Hash Min Len Max Len Index Chain Len Im 1 7 0 2400	Chain Count N* of tables	
Charset		
alpha-numeric	Edit	
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789		
Table properties		
Key space: 80603140212 keys		
Disk space: 2.98 GB (610.35 MB each table)		
Success probability: 0.990635 (99.06%)		
Benchmark		
Hash speed: 689464 hash/sec		
Step speed: 481046 step/sec		
Table precomputation time: 2.30978 days		
Total precomputation time: 11.5489 days		
Max cryptanalysis time: 29.9348 seconds		

Figure 4.3: Creating 5 rainbow tables with a success rate of 99.06% using about 3 GB of space.



Here is a handy reference table:

There are some typical configurations (for LM hash type, length from 1 to 7) you can use, for example:

	#1	#2	#3	#4
Charset	alpha	alpha-numeric	alpha-num-sym14	all
Chain length	2,100	2,400	12,000	20,000
Chain count	8,000,000	40,000,000	40,000,000	100,000,000
Tables	5	7	13	20
Success rate	99.9%	99.9%	99.9%	99,6%
Total space	640 Mb	4,480 Mb	8,320 Mb	32,000 Mb
Max gen. time	18h 35m	6d 5h	67d 18h	369d
Max analysis time	8 s	16 s	15 m	53 m

Example 5:

Using Cain and Abel to crack passwords using Rainbow Tables

Step 1: Download⁸ and install Cain.

Step 2: Click on the "Cracker" tab. Select what type of passwords you want to crack. In this case LM & NTLM Hashes. Then right click and select "add to list." Navigate to where you have your text file of hashes, select it and then select next.

File View Configure T	ools Help			
🔄 🕸 🙃 🛛 🕇 🗑 👒	P. 1 m m 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	۶ 🕥 😁 🛿 🍭	1.	
🎯 Protected Storage 💡 Net	vork 🏟 Sniffer 🎯 LSA Secrets 🥑 Cracker 😨	Traceroute 🔝 CCDU	Wireless	
🥑 Cracker 🛛 🔥 User	Na Add NT Hashes from	LM Hash	NT Hash	
LM & NTLM Hash MTLMv2 Hashes (0) MS-Cache Hashes (0) Gisco IOS-MD5 Hash Cisco IOS-MD5 Hash Gisco PIX-MD5 Hash APOP-MD5 Hashes OSPF-MD5 Hashes VRR-HMAC Hashes WRR-HMAC Hashe: WRC-MD5(0) MD2 Hashes (0) MD2 Hashes (0) MD5 Hashes (0)	Import Hashes from local system Include Password History Hashes Include Password			
- SHA SHA-1 Hashes (0) - SHA SHA-2 Hashes (0) - RIPEMD-160 Hashe - Kerb5 PreAuth Hash	Cancel N	ext→		
Radius Shared-Key			>	
M55QL Hashes (0)	LM & NTLM Hashes			
http://www.oxid.it			11.	

Figure 5.1: Loading hashes from file

⁸ Download Cain and Abel from: <u>http://www.oxid.it/</u>

II Security O	nline, Inc. ©			\checkmark	<u>https://</u>	www.learns	securityonline.
		logro	Ca	ourity	Inlino		
		LCUIII	J	curity	UIIIIIC		
1							
File View Config	jure Tools Help						
a 🙃 🕇 🗑	B. 8. 1	📟 🅐 🚾 🛲		°a 🕵 🗶 💋	2 6 ?	i	
A A L A	64 1 1			UL 45 89 CA	- v :]	. L	-
otected Storage	👂 Network 🛛 🏟 Sniffer	B LSA Secrets	🕜 Cra	cker 🔯 Tracerou	te 🔝 CCDU 🧖	Wireless	
ker 🔨	User Name	LM Password	< 8	NT Password	LM Hash	NT Hash	
M & NTLM Hash	X testuser1	LIM Password	1 < 0	INT Password	0F20048EFC64	1120ACB74670	
					UF20048EFC64	112040674670	
TLMv2 Hashes (0)							
사람이 아파는 것 같은 것에 주셨는데.	X remote				E52CAC67419A	8846F7EAEE8F	
IS-Cache Hashes (🗙 joeuser				E52CAC67419A	8846F7EAEE8F 8846F7EAEE8F	
IS-Cache Hashes (WL files (0)	X joeuser X averageguy		*		E52CAC67419A 299CCF964D9A	8846F7EAEE8F 8846F7EAEE8F A5C07214487C	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl	X joeuser X averageguy X harderpass		*		E52CAC67419A 299CCF964D9A B75838F7A57E	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC50F8A8149C	
15-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hasl	X joeuser Averageguy Aarderpass Ademouser				E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC50F8A8149C 371D5760453C	
15-Cache Hashes (WL files (0) iisco IOS-MD5 Hasl iisco PIX-MD5 Hast POP-MD5 Hashes	 joeuser averageguy harderpass demouser randy 		*		E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44 98B5AFEB6729	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC50F8A8149C 371D5760453C A9F34664151F	
15-Cache Hashes (WL files (0) iisco IOS-MD5 Hasl iisco PIX-MD5 Hast POP-MD5 Hashes IRAM-MD5 Hashes	 joeuser averageguy harderpass demouser randy Asmith 		*		E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44 98B5AFEB6729 E165F0192EF8	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC50F8A8149C 371D5760453C A9F34664151F E4E8E0E7EF70.	
ITLMv2 Hashes (0) IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hash isco PIX-MD5 Hashes iRAM-MD5 Hashes iSPF-MD5 Hashes =	X joeuser X averageguy X harderpass X demouser X randy X Asmith X Bsmith		*		E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44 98B5AFEB6729 E165F0192EF8 136A8418CF76	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC50F8A8149C 371D5760453C A9F34664151F E4E8E0E7EF70, 3431E75AD08D	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hash POP-MD5 Hashes RAM-MD5 Hashes ISPF-MD5 Hashes (IPv2-MD5 Hashes RRP-HMAC Hashes	joeuser averageguy harderpass demouser randy randy Asmith Bsmith csmith		*		E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44 98B5AFEB6729 E165F0192EF8 136A8418CF76 BB26C0635328	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC505F8A8149C 37105760453C A9F34664151F E4E8E0E7EF70, 3431E75AD08D A2746ED41299	
IS-Cache Hashes (WL files (0) isco 105-MD5 Hasl isco PIX-MD5 Hashes POP-MD5 Hashes SPF-MD5 Hashes SPF-MD5 Hashes IPV2-MD5 Hashes RAP-HMAC Hashe NC-3DE5 (0)	joeuser averageguy harderpass demouser randy Asmith Ssmith csmith Dsmith		*		E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44 98B5AFEB6729 E165F0192EF8 136A8418CF76 B826C0635328 A8EED815A197	8846F7EAE88F 8846F7EAE88F A5C07214487C EC50F8A8149C 37105760453C A9F34664151F E4EBE0E7EF70. 3431E75AD08D A2746ED41299 F09A31889C35	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hasl Scor PIX-MD5 Hashes POP-MD5 Hashes SPF-MD5 Hashes I IPV2-MD5 Hashes RRP-HMAC Hashe: NC-3DE5 (0) ID2 Hashes (0)	joeuser averageguy harderpass demouser randy randy Asmith Bsmith csmith		*		E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44 98B5AFEB6729 E165F0192EF8 136A8418CF76 BB26C0635328	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC505F8A8149C 37105760453C A9F34664151F E4E8E0E7EF70, 3431E75AD08D A2746ED41299	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hashes isco PIX-MD5 Hashes RRM-MD5 Hashes I IPV2-MD5 Hashes RRP-HMAC Hashes NC-3DE5 (0) ID2 Hashes (0) ID4 Hashes (0)	joeuser averageguy harderpass demouser randy Asmith Bsmith csmith Dsmith Esmith Esmith Esmith		*		E52CAC67419A 299CCF964D9A B75838F7A57E 261A6631FE44 98B5AFEB6729 E165F0192EF8 136A8418CF76 B826C0635328 A8EED815A197	8846F7EAE88F 8846F7EAE88F A5C07214487C EC50F8A8149C 37105760453C A9F34664151F E4EBE0E7EF70. 3431E75AD08D A2746ED41299 F09A31889C35	
15-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hashes ISOP-MD5 Hashes ISOP-MD5 Hashes ISOP-MD5 Hashes ISOP-MD5 Hashes IRP-HMAC Hashe INC-3DE5 (0) ID2 Hashes (0) ID4 Hashes (0)	joeuser averageguy harderpass demouser randy Asmith Bsmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith		*		E52CAC67419A 299CCF964D9A 87838F7A57E 261A6631FE44 985AFE66729 E165F0192EF8 136A8418CF76 B826C0635328 A8EE0815A197 5A9D89F88B5D	8846F7EAE88F 8846F7EAE88F A5C07214487C ECS0F8A8149C 37105760453C A9F34664151F E4E8E0E7EF70. 3431E75AD08D A2746ED41299 F09A31889C35 5FCC20A69EC7	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hashes ISCO PIX-MD5 Hashes SPF-MD5 Hashes SPF-MD5 Hashes IPV2-MD5 Hashes IPV2-MD5 Hashes IPV2-MD5 Hashes INC-3DE5 (0) ID2 Hashes (0) ID4 Hashes (0) HA-1 Hashes (0)	joeuser averageguy harderpass demouser randy Asmith Bsmith Csmith Csmith Esmith Esmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith		*		E52CAC67419A 299CCF964D9A B78838F7A57E 261A6631FE44 9885AFE66729 E165F0192EF8 136A8418CF76 B826C0635328 A8EED815A197 SA9D89F6885D 213D466D8582	8846F7EAE88F 8846F7EAE88F ASC07214487C EC50F8A8149C 37105760453C A9F34664151F E4E8E0E7EF70. 3431E75A008D A2746E041299 F09A31889C35 SFCC20A69EC7 FAF10460760F	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hashes isco PIX-MD5 Hashes iRAM-MD5 Hashes IPV2-MD5 Hashes IPV2-MD5 Hashes IPV2-MD5 Hashes IPV2-MD5 Hashes (0) ID2 Hashes (0) ID5 Hashes (0) ID5 Hashes (0) ID5 Hashes (0) ID4-1 Hashes (0) HA-2 Hashes (0)	joeuser averageguy harderpass demouser randy Asmith Bsmith Ssmith Ssmith Esmith Esmith Esmith Ssmith Ssmith Ssmith		* * *		E52CAC67419A 299CCF964D9A B78838F7A57E 261A6631FE44 9885AFE86729 E165F0192EF8 136A8418CF76 B826C063528 A8EE0815A197 243D89F88850 213D4660B582 385A83A746BF	8846F7EAE88F 8846F7EAE88F ASC07214487C 25076848149C 371D5760453C A9F34664151F E4E8E0E7EF70. 3431E75AD08D A2746ED41299 F09A31889C53 FFC20A69EC7 FAF10460760F 1CC1B3958856	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hashes ISOP-MD5 Hashes RAM-MD5 Hashes ISPF-MD5 Hashes ISPF-MD5 Hashes RRP-HMAC Hashes NC-3DE5 (0) ID2 Hashes (0) ID4 Hashes (0) ID5 Hashes (0) HA-1 Hashes (0) HA-2 Hashes (0) IHA-2 Hashes (0) IHA-1 Hashes (0) IHA-2 Hashes (0) IHA-2 Hashes (0) IHA-2 Hashes (0) IHA-1 Hashes (0) IHA-2 Hashes (0) IHA-2 Hashes (0) IHA-1 Hashes (0) IHA-2 Hashes (0) IHA-2 Hashes (0) IHA-2 Hashes (0) IHA-2 Hashes (0) IHA-1 Hashes (0) IHA-2 HASHE IHA-2 HASHE IHA-	joeuser averageguy harderpass demouser randy Asmith Bsmith Csmith Csmith Esmith Esmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith Ssmith		* * *		E52CAC67419A 299CCF964D9A B78838F7A57E 261A6631FE44 9885AFE86729 E165F0192EF8 136A8418CF76 B826C0635328 A8EED815A197 5A9DB9F88B5D 213D466D8582 385A83A746BF 788CCAEE08C9	8846F7EAEE8F 8846F7EAEE8F A5C07214487C EC50F8A8149C 37105760453C A9F34664151F E4EBE0E7EF70. 3431E75AD08D A2746ED41299 F09A31889C35 SFCC20A69EC7 FAF10460760F 1CC1B3958856 972E8E7D5568	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hasl Sico PIX-MD5 Hashes POP-MD5 Hashes ISPF-MD5 Hashes ISPF-MD5 Hashes IPV2-MD5 Hashes IPV2-MD5 Hashes IPV2-MD5 Hashes (0) ID2 Hashes (0) ID5 Hashes (0) ID5 Hashes (0) ID4 Hashes (0) ID5 Hashes (0) ID5 Hashes (0) ID4 Hashes (0) ID5 Hashes (0)	joeuser averageguy harderpass demouser candy Asmith Bsmith Ssmith Ssmith Esmith Esmith Ssmith		* * *		E52CAC67419A 299CCF964D9A B78838F7A57E 261A6631FE44 985AFE6529 E165F0192EF8 136A8418CF76 B826C0635328 A8EE0815A197 5A9DB9F88B5D 213D466DB582 78BCCAEE08C9 59E2D865E9D4	8846F7EAEE8F 8846F7EAEE8F A5C07214487C ECS0F8A8149C 37105760453C A9F34664151F E4EBE0E7EF70. 3431E75AD080 A2746ED41299 F09A31889C35 5FCC20A69EC7 FAF10460760F 1CC183958856 972E82705568 147D125645D4	
IS-Cache Hashes (WL files (0) isco IOS-MD5 Hasl isco PIX-MD5 Hashes isco PIX-MD5 Hashes RRM-MD5 Hashes I IPV2-MD5 Hashes RRP-HMAC Hashes NC-3DE5 (0) ID2 Hashes (0) ID4 Hashes (0)	joeuser averageguy harderpass demouser randy Asmith Bsmith Ssmith Ssmith		* * * * *		E52CAC67419A 299CCF964D9A B78838F7A57E 261A6631FE44 9885AFE66729 E165F0192EF8 136A8418CF76 B826C0635329 A8EED0615A197 5A9DB9F88B5D 213D466DB582 385A83A746BF 788CCAEE08679 59E2D885E9D4 59E2D885E9D4	8846F7EAEE8F 8846F7EAEE8F ASC07214487C ECS0F8A8149C 37105760453C A9F34664151F E4E8E0E7EF70. 3431E75AD08D A2746ED41299 F09A31889C35 SFCC20A69EC7 FAF10460760F 1CC183958856 972E8E7D5568 147D125645D4 147D125645D4	

Figure 5.2: Hashes loaded into Cain, ready to be cracked.

Step 3: Right click and select "select all" then right click again and select cryptanalysis attack and "LM Hashes via RainbowTables"

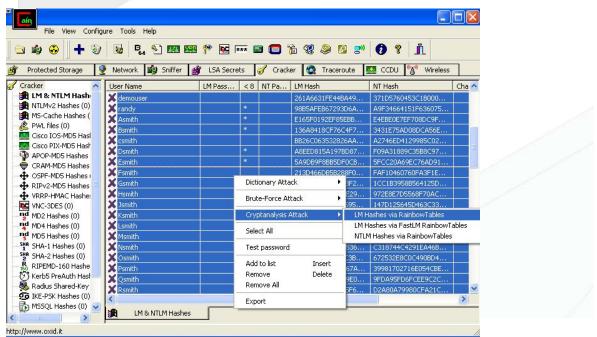


Figure 5.3: Selecting a cryptanalysis attack via RainbowTables.

Step 4: Click on Add Table. Then navigate to where you have your rainbow tables, highlight them all and select Open.

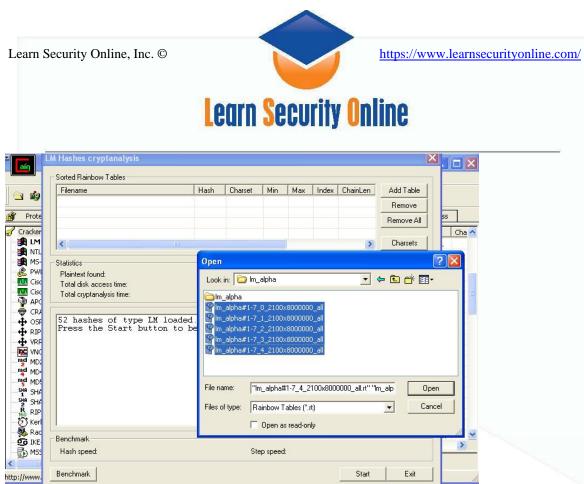


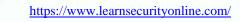
Figure 5.4: Adding your rainbow tables to use for cracking.

Step 5: Click on "Start" and Cain will start to work through the rainbow tables.

Filename	Hash	Charset	Min	Max	Index	ChainL(🔨	Add Table	
D:\torrents\Im_alpha\Im_alpha#1-7_0	lm	alpha	1	7	0	2100	Remove	
🐉 D:\torrents\lm_alpha\lm_alpha#1-7_1	Im	alpha	1	7	1	2100 🦲		
D:\torrents\lm_alpha\lm_alpha#1-7_2	Im	alpha	1	7	2	2100	Remove All	ss
D:\torrents\lm_alpha\lm_alpha#1-7_3	Im	alpha	1	7	3	2100 💌		h 🔼
<						>	Charsets	B74670
itatistics								'EAEE8F
		т.						'EAEE8F
Plaintext found: Total disk access time:			tal chain tal false		p:			214487C
Total disk access time: Total cryptanalysis time:			tal false		-			A8149C
r otar cryptanaysis tine.		10	(al laise	aiainii ste	μ.			760453C
								64151F
Reading lm_alpha#1-7_0_2100:	x80000	00_all.	rt	2			~	E7EF70.
119803904 bytes read in Verifying the file (OK)	: 31.1	5 5						SAD08D
Searching for 41 hashes								D41299
Plaintext of e52cac67419a9a							_	.889C35
Plaintext of 4a3b108f3fa6cb								DA69EC7
Plaintext of b75838f7a57ee6 Plaintext of 261a6631fe44ba	/9 18 49 ie	DEMODYC						60760F
Plaintext of 98b5afeb67293d	6a is	RANDY						3958856
Plaintext of e165f0192ef85e								7D5568
Plaintext of 385a83a746bfa8 Plaintext of 13d855fc4841c7		GHGHGH ABCDEF						25645D4
Plaintext of d71808bf36f815		FOOTBAL						5645D4
D1-1-++ -+ 0-024-40(020		COCCER						C92C0E.
								4896575 💙
enchmark								

Figure 5.5: Cain working through the Rainbow Tables cracking passwords.

Step 6: When its all done click Exit and it will show you the cracked passwords.



Learn Security Online

Filename		Hash	Charset	Min	Max	Index	ChainL 🔨	Add Table
VD:\torrents\Im_alpha\Im_al	pha#1-7_2	Im	alpha	1	7	2	2100	Remove
VD:\torrents\Im_alpha\Im_al	pha#1-7_3	Im	alpha	1	7	3	2100	
✓D:\torrents\Im_alpha\Im_al	pha#1·7_4	lm	alpha	1	7	4	2100	Remove All
<							>	Charsets
Statistics								
Plaintext found: 26 of 41 (63.4						p: 20574	3101	
Total disk access time: 131.00	2.57		1.7		alarm: 19			
Total cryptanalysis time: 735.4	6 s		Ic	tal false -	alarm ste	p: 143323	/913	
Lsmith	ABCDEF							1
	notfound							-
	FOOTBAL <n SOCCER</n 	otfour	nd>					
	LSOISDABE	ST						
testuser1 '	FESTUSE <n< td=""><td>otfour</td><td>nd></td><td></td><td></td><td></td><td></td><td></td></n<>	otfour	nd>					
	RAINBOWTA	BLEZ						
	CROKET							
	notfound	>						
	(notfound							
joeuser l	PASSWORD							
								>
Benchmark								
Hash speed:				ep speed				

Figure 5.6: Cain finishes running though the Rainbow Tables.

🕼 🚱 🛛 🕂 🥹 🐼 🖪 🗧 🕥 Ditected Storage	Sniffer 🔐 LSA Secrets		🗖 📸 🕵 🧶 Cracker 🔯 Trace	🖄 🛃 🚺 😵	Wireless
ker 🔥 User Name	LM Password	< 8	NT Password	LM Hash	NT Hash
M & NTLM H	GHGHGH	*	gHgHgH	385A83A746BF	1CC1B3958B56.
ITLMv2 Hashes Kennith	777777	*		78BCCAEE08C9	972E8E7D5568.
15-Cache Hash 🗙 Jsmith				59E2DB85E9D4	147D125645D4.
WL files (0) Ksmith				59E2DB85E9D4	147D125645D4.
Sisco IOS-MD5	ABCDEF	*	ABCdef	13D855FC4841	3DCEBC92C0E.
isco PIX-MD5 I	FOOTBAL??????			D71808BF36F8	45E8DA896575
RAM-MD5 Has Nsmith	SOCCER	*	SOCCER	9C92FA4960AC	C318744C4291
SPF-MD5 Has	CROKET	*	CROKET	1153C3961EE5	672532E8C0C4
IPv2-MD5 Has SPsmith	7777777	*		4A01C0E45FCA	39981702716E.
RRP-HMAC He RQsmith	HOWNOW	*	HOWNOW	6842A19CC4C5	9FDA95FD6FCE
NC-3DES (0) Rsmith	BROWNCOW		BROWNCOW	BC472F3BF9A0	D2A80A79980C
D2 Hashes (0 🕺 💦 Ssmith	GHANDI	*	gHaNdI	09755C01D278	62F740C2EA31
04 Hashes (0 🛛 🐴 Tsmith	ABCDEF	*	ABCdef	13D855FC4841	3DCEBC92C0E.
05 Hashes (0 Smith	RTDOTNET		RTdotnet	9E2204E2058A	476541DEC5CB
HA-1 Hashes (🛛 🗙 Vsmith				7F9CD2D7C934	16FAABB24B95
IA-2 Hashes (🛛 😤 Wsmith	EASYONEISNTIT		EASYoneISNTit	AC814111DF80	15B194EB8D8F.
PEMD-160 Ha 🗙 Xsmith	* empty *	*		AAD3B435B514	2321504F2FA9.
erb5 PreAuth	LSOISDABEST		LSOISDABEST	D5662E6B2365	75344B75B5A9.
adius Shared-	RAINBOWTABLEZ		RAINBOWTABLEZ	9224FC255C58	0C105C9F4326
KE-PSK Hashe:					

ttp://www.oxid.it

Figure 5.7: Our cracked passwords in Cain. Notice that Cain also found the NTLM password based on the LM password.



I am still confused what does "X, Y, or Z" mean?

Here are some things that may not be immediately clear when dealing with rainbow tables:

1- What does "t", "m", and "l" mean or stand for?

To answer this, let's analyze an rtgen command:

rtgen lm alpha 1 7 0 2100 8000000 all

rtgen obviously means the program to run. "**Im**" means we want to generate LAN Manager tables. "**alpha**" mean we want to use the characters listed in our charset.txt file for alpha: alpha = [ABCDEFGHIJKLMNOPQRSTUVWXYZ]

"1" and "7" are our plaintext ranges. So we want passwords from "A" to "ZZZZZZZ." If we had put plaintext length range "4-6", "AAAA" and "ZZZZZZ" would be among the key space; but "AAA" would not because it has a length 3. Remember that, for LAN Manager , passwords they are broken up into 7 character chunks, so there would be no need to do a plaintext range of 1 to 8. The "0" is our table number or rainbow **table count**, if you look at the rtgen commands to generate configuration #0 we create five tables 0 to 4. This is so we can split up tables between computers making the rainbow tables and to increase our success rate. "2100" is our rainbow **chain length**. Chain length increases the success rate per table but does not increase table size. It computes more hashes per chain but also takes longer to create and search the table. A common "upper" value for chain length is 4000-5000. "8000000" is our rainbow **chain count** of each rainbow table. Chain count is simply how many chains you want per table. Increasing this value produces larger files with higher success rates, but the overall computation time isn't affected. You can adjust the chain count so your rainbow tables are conveniently sized (like for a CD or DVD). The "all" is our file title suffix or what we want appended to the end of our table's file name, it can be anything you want.

2- What do the different chain lengths and chain counts mean?

Chain Length increases the success rate per table. It computes more hashes per chain but also takes longer to create and search the table. A common "upper" value for chain length is 4000-5000 but it can be whatever you want. Chain count is simply how many chains you want per table. Increasing this value produces larger files with higher success rates, but the overall computation time isn't affected. You can adjust the chain count so your rainbow tables are conveniently sized (like for a CD or DVD) or to increase the success rate.

3- Why can't I create just one rainbow table?

You can! But to get a high enough success rate that table will be too large to search in a reasonable amount of time. That is why we normally create several. Now we could, by adjusting chain length and chain count, create a giant rainbow table but we will have to sort it, which will take a long time and then search it, which will take an even longer time; thus reducing the whole point of rainbow tables. It would be a more efficient use of space to create many rainbow tables so you can sort and search them faster.

4-I am still confused!

Then 1) go read the paper: <u>http://lasecwww.epfl.ch/php_code/publications/search.php?ref=Oech03</u> and 2) check out the next section for some examples with Winrtgen which allows you to see (graphically) how changing values changes success rates, table size, and table generation time.



Using Winrtgen to see how chain length, chain count and number of tables effects success rate and computation time

We can see in this example that we get a 97.80% success rate with one LM ALPHA rainbow table with a Chain Length of 2400 and a Chain Count of 40,000,000. It will take 2.23 days to generate the table on the computer (a P3 1.0 Ghz with 512 MB of RAM).

nrtgen v1.8 (Rainbow Tables Generator) by mao	×
Rainbow Table properties	-
Hash Min Len Max Len Index Chain Len Chain Count N° of tables 1 Im 💽 1 7 0 2400 40000000 1	
Charset	
alpha 🗾 Edit	
ABCDEFGHIJKLMNOPQRSTUVWXYZ	
Table properties Key space: 8353082582 keys Disk space: 610.35 MB Success probability: 0.978038 (97.80%)	
Benchmark	>
Hash speed: 715307 hash/sec	
Step speed: 496228 step/sec	
Table precomputation time: 2.23911 days	-
Total precomputation time: 2.23911 days	
Max cryptanalysis time: 5.80378 seconds	
Benchmark OK Cancel	

Increasing the Chain Length to 4000 increases our success rate to 99.11% but it now takes 3.67 days to generate the table.

Vinrtgen v1.8 ((Rainbow Tal ble propertie		tor) by r	mao					×	
Hash	Min Len-	Max Len	Index 0	Chain Len 4000	Chain Cou 40000000		N° of tables			
Charset				an Ca	- 5					
alpha						_	Edit		- C - 5	
ABCDEFGH	IJKLMNOPQRS	TUVWXYZ							1	
Disk space:	8353082582 key									
Benchmark	-							>		
Hash speed	: 717360 hash/s									
-	504337 step/se									
	mputation time: 3 mputation time: 3									
	nputation time: 5.86 halysis time: 15.86	an an an an an Garage								
Benchmark	1					ОК	Cancel			



rtgen v1.8 (Rainbow Tables Generator) by mao		_	
ainbow Table properties		Þ	
	ain Count	N* of tables	
Charset		1	
alpha	•	Edit	
ABCDEFGHIJKLMNOPQRSTUVWXYZ			
Table properties			
Key space: 8353082582 keys			
Disk space: 122.07 MB			
Success probability: 0.751327 (75.13%)			
Benchmark			
Hash speed: 721500 hash/sec			
Step speed: 507408 step/sec			
Table precomputation time: 9.19707 hours			
Total precomputation time: 9.19707 hours			
Max cryptanalysis time: 4.34562 seconds			
Benchmark	ОК	Cancel	

LM Configuration #0 with 5 tables (the recommended configuration). Notice that for roughly the same amount of time and space as our first example with a Chain Length of 2400 and a Chain Count of 40,000,000 and a success rate of 97.80% we can get 99.90% with this Rainbow Table configuration. Another thing to note that we don't see is sort time and how much longer it takes to sort one big table versus several smaller tables.

F Re	inbow Table properties 🛛 🕹		
	Hash Min Len Max Len Index Chain Len Chain Count N° of tables Im 1 7 0 2100 8000000 5		
	Charset		
	alpha 🗾 Edit		
	ABCDEFGHJKLMNOPQRSTUVWXYZ		
	Table properties		
	Key space: 8353082582 keys		
	Disk space: 610.35 MB (122.07 MB each table)		
	Success probability: 0.999049 (99.90%)		
	Benchmark	15	
Ì	Hash speed: 713266 hash/sec		
1	Step speed: 481881 step/sec		
	Table precomputation time: 9.68427 hours		
	Total precomputation time: 2.01756 days		
	Max cryptanalysis time: 22.8791 seconds		

Let's see how long it takes to create tables to find "all" possible password combinations—minus "ALT-XXX commands." For a 1GB table it will take 23 days with a 12.25% success rate.



And it will take 4.6 years (on a P3 1GHz machine) to generate enough tables to reach 99.98% !!!

Hash Min Len Max Len Index Im 💽 1 7 1	Chain Len Chain Count N° of tables 15200 67108864 64	
Charset	· · · · · · · · · · · · · · · · · · ·	
all-space	Edit	
ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789(@	ù#\$%^&*∩- +=~`∏{\\.'"<> ?/	
Table properties		
Key space: 7555858447479 keys		
Disk space: 64 GB (1 GB each table)		
Success probability: 0.999766 (99.98%)		
Benchmark		
Hash speed: 717360 hash/sec		
Step speed: 447387 step/sec		
Table precomputation time: 26.3892 days		
Total precomputation time: 4.62715 years		
Max cryptanalysis time: 4.59041 hours		

Protecting yourself against RainbowCrack attacks and other password attacks

-Limiting physical access

-Continue to force the use of special characters

-Use ALT-XXX characters in your passwords



-Keep up with updates -Use Pass phrases -Use Multi-factor authentication -Password Change Policy -Use NTLM or NTLMv2

What if my Windows password is longer than 14 characters or the LAN Manager hash is not stored?

If your systems do not require the LAN Manager (LM) hash (for example if you are running an Active Directory (AD) domain), or if your password is >14 characters long, the LM hash will be stored as the blank LM hash, even though the clear text password itself is not blank. Basically none of the cracking tools will see a LM hash.

If this is the case, you will need to audit your password hashes against the NTLM character set.

Limiting physical access

It's a well-known fact that if someone has physical access to a machine then it's not secure. They can walk off with it, take the hard drive, turn it off, etc. One common attack if you have physical access to a machine is to use a bootable Linux distro to simply boot into Linux and grab the SAM file off the windows partition. You can then crack it at your leisure. IronGeek wrote a good tutorial on this method and even has a video you can watch. You can get it here: <u>http://www.irongeek.com/i.php?page=security/localsamcrack2</u>. Another interesting tool released by Eeye is SysRQ2: <u>http://research.eeye.com/html/tools/RT20060801-8.html</u>. "SysRq is a bootable CD image that allows a user to open a fully privileged (SYSTEM) command prompt on Windows 2000, Windows XP, and Windows Server 2003 systems by pressing Ctrl+Shift+SysRq at any time after startup."

Continue to force the use of special characters

Even though rainbow tables can rip thru a LM password with any type of special character it still takes a large amount of time (1-2 years) to generate them, this will deter most people or force them to use an online hash cracking service⁹. It also greatly increases the time for brute force attempts. In LC4 we go from 9-11 **hours** to brute force alpha-numeric password to 91 **days** to brute force passwords with the possibility of all special characters (minus ALT-XXX passwords).



⁹ <u>http://www.rainbowcrack-online.com/</u> or <u>http://www.plain-text.info/</u>



Brute-force cracking time is greatly increased by using special characters in your passwords. From 9 hours with just alpha-numeric to 91 days with all characters.

Use ALT-XXX characters in your passwords

ALT characters are produced by holding down the ALT key (or FN-ALT keys on some laptops) and pressing a three or four digit number sequence on your keypad. Most password crackers cannot crack passwords with ALT characters. Most ALT characters also have the added benefit that passwords that have ALT characters in them cannot be stored as LM hashes.

ALT characters which cause the LMHash to disappear

0128-0159	0306-0307	312	0319-0320	
0329-0331	383	0385-0406	0408-0409	
0411-0414	0418-0424	426	0428-0429	
0433-0437	0439-0447	0449-0450	0452-0460	
477	0480-0483	0494-0495	0497-0608	
0610-0631	0633-0696	699	0701-0707	
709	711	716	0718-0729	
731	0733-0767	0773-0775	777	
0779-0781	0783-0806	0808-0816	0819-0893	
0895-0912	914	0918-0919	0921-0927	
0929-0930	933	0935-0936	0938-0944	
947	0950-0955	0957-0959	0961-0962	
965	0967-1024			

Some ALT characters not listed above, firstly, can still be stored as LM Hashes and secondly, can weaken your password because they are converted to uppercase before they are stored. If you are going to use ALT characters go with the "safe" ones above.



A final note to add about ALT characters is that you may be unable to login to mixed environments with ALT passwords or be unable to authenticate to file shares using SAMBA or other non-Windows tools.

Keep up with updates

Keep up with your security patches. While you can't protect against zero day exploits you can protect against exploits that have patches! All of the password dumping tools must have *administrative* level privileges to dump the hashes. You can keep the majority of the bad guys out by patching your machines promptly against public exploits. This will help keep you protected from that system/administrative level exploit that was just released to the public.

Use Pass phrases

Using pass phrases is the easiest and simplest way to protect you network from password cracking. If your password policy makes use of pass phrases that are greater than 14 characters AND use special characters you can protect yourself from all but the determined attackers. If your network is Windows 2000 and above you have a maximum length of 127 characters on your password/pass phrase; so sky's the limit. A pass phrase like "This is my Stupid Pass Phrase!" is long enough to be stored as NTLM or NTLMv2 (because it is longer than 14 characters), has Uppercase, Lowercase, Spaces, and Special Characters, and is easy to remember. This is a much more secure password than even "@w3cjd\$Beu=mDr". If you can get your users to do some character substitution on their pass phrases even better!

The use of strong passwords within an environment needs to be mandated for users. Using the stronger NTLMv2 hashing scheme won't prevent a successful dictionary attack. The use of strong passwords can be enforced on Windows NT through the use of the passfilt.dll. This is described in Microsoft Knowledgebase Article 161990¹⁰. The use of strong passwords in Windows 2000, XP and 2003 can be enforced by settings in the Group Policy, which is described in Microsoft Knowledgebase Article 225230¹¹.

Use Multi-factor authentication

Any decent CISSP could talk at length about multi-factor authentication and many have, so I won't cover it much here except to say that any type of multi-factor authentication you can implement will help your network. Whether it be biometric, smartcard, token, etc, anything that makes a user need to provide more than a simple password to log on to your network helps your security posture and stops a cracked password in its tracks.

Password Change Policy

A strong password policy will:

* Insist on frequent password changes (frequency depends on the sensitivity of your data anywhere between 30-180 days)

* Require long passwords composed of random combinations of upper and lowercase letters, numbers and special characters

- * Not allow blank passwords
- * Check to ensure passwords are not repeated
- * Prevent the use of any part of the user's name or user ID
- * Not allow the use of common dictionary words

On Windows 2000 or above with Active Directory, password management is fairly easily controlled and implemented using group policy. Forcing users to meet your password policy requirements is fairly easy with Active Directory and should be utilized if possible.

¹⁰ <u>http://support.microsoft.com/default.aspx?scid=kb;en-us;161990</u>

¹¹ <u>http://support.microsoft.com/default.aspx?scid=kb;en-us;225230</u>



Use NTLM or NTLMv2

Instead of storing your user account password in clear-text, Windows generates and stores user account passwords by using two different password representations, generally known as "hashes." When you set or change the password for a user account to a password that contains fewer than 15 characters, Windows generates both a LAN Manager hash (LM hash) and a Windows NT hash (NT hash) of the password. These hashes are stored in the local Security Accounts Manager (SAM) database or in Active Directory.

The LM hash is relatively weak compared to the NT hash, and it is therefore prone to fast brute force attack. Therefore, you may want to prevent Windows from storing an LM hash of your password

Windows 2000-based servers and Windows Server 2003-based servers can authenticate users who connect from computers that are running all earlier versions of Windows. However, versions of Windows earlier than Windows 2000 do not use Kerberos for authentication. For backward compatibility, Windows 2000 and Windows Server 2003 support LAN Manager (LM) authentication, Windows NT (NTLM) authentication, and NTLM version 2 (NTLMv2) authentications. The NTLM, NTLMv2, and Kerberos all use the NT hash, also known as the Unicode hash. The LM authentication protocol uses the LM hash. The use of LAN Manager hashes on the network can be disabled on Windows NT, 2000, 2003 & XP through registry edits or through the Local Security Policy. The instructions to do so can be found at in Microsoft Knowledgebase Article 147706¹². The storage of LAN Manager hashes also needs to be disabled; this can be done for Windows 2000, XP and 2003 again via registry edits or the Local Security Policy. The instructions to do so can be found at in Microsoft Knowledgebase Article 299656¹³.

Method 1: Implement the NoLMHash Policy by Using Group Policy

To disable the storage of LM hashes of a user's passwords in the local computer's SAM database by using Local Group Policy (Windows XP or Windows Server 2003) or in a Windows Server 2003 Active Directory environment by using Group Policy in Active Directory (Windows Server 2003), follow these steps:

- 1. In Group Policy, expand Computer Configuration, expand Windows Settings, expand Security Settings, expand Local Policies, and then click Security Options.
- 2. In the list of available policies, double-click Network security: Do not store LAN Manager hash value on next password change.
- 3. Click **Enabled**, and then click **OK**.

Method 2: Implement the NoLMHash Policy by Editing the Registry

Windows 2000 SP2 and Later

To add this key by using Registry Editor, follow these steps:

- 1. Start Registry Editor (Regedt32.exe).
- 2. Locate and then click the following key:
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa
- 3. On the Edit menu, click Add Key, type NoLMHash, and then press ENTER.
- 4. Quit Registry Editor.
- 5. Restart the computer, and then change your password to make the setting active.

Notes

¹² http://support.microsoft.com/default.aspx?scid=kb;en-us;147706

¹³ <u>http://support.microsoft.com/default.aspx?scid=KB;EN-US;q299656&</u>



- This registry key change must be made on all Windows 2000 domain controllers to disable the storage of LM hashes of users' passwords in a Windows 2000 Active Directory environment.
- This registry key prevents new LM hashes from being created on Windows 2000-based computers, but it does not clear the history of previous LM hashes that are stored. Existing LM hashes that are stored will be removed as you change passwords.

Windows XP and Windows Server 2003

- 1. Click **Start**, click **Run**, type **regedit**, and then click **OK**.
- 2. Locate and then click the following key in the registry:
- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa
- 3. On the Edit menu, point to New, and then click DWORD Value.
- 4. Type **NoLMHash**, and then press ENTER.
- 5. On the Edit menu, click Modify.
- 6. Type 1, and then click OK.
- 7. Restart your computer, and then change your password.

Notes

- This registry change must be made on all Windows Server 2003 domain controllers to disable the storage of LM hashes of users' passwords in a Windows 2003 Active Directory environment. If you are a domain administrator, you can use Active Directory Users and Computers Microsoft Management Console (MMC) to deploy this policy to all domain controllers or all computers on the domain as described in Method 1 (Implement the NoLMHash Policy by Using Group Policy).
- This DWORD value prevents new LM hashes from being created on Windows XP-based computers and Windows Server 2003-based computers. The history of all previous LM hashes is cleared when you complete these steps.

Windows NT

Control of NTLM security is through the following registry key: HKEY_LOCAL_MACHINE\System\CurrentControlSet\control\LSA

Name: LMCompatibilityLevel

Type: REG_DWORD

Value: 5 : DC refuses LM and NTLM responses (accepts only NTLMv2)

Value: 4 : DC refuses LM responses

Value: 3 : Send NTLMv2 response only

Value: 2 : Send NTLM response only

Value: 1 : Use NTLMv2 session security if negotiated

Value: 0 : default - Send LM response and NTLM response; never use NTLMv2 session security

More information on the values:

Level 0 - Send LM and NTLM response; never use NTLM 2 session security. Clients use LM and NTLM authentication, and never use NTLM 2 session security; domain controllers accept LM, NTLM, and NTLM 2 authentication.

Level 1 - Use NTLM 2 session security if negotiated. Clients use LM and NTLM authentication, and use NTLM 2 session security if the server supports it; domain controllers accept LM, NTLM, and NTLM 2 authentication.



Level 2 - Send NTLM response only. Clients use only NTLM authentication, and use NTLM 2 session security if the server supports it; domain controllers accept LM, NTLM, and NTLM 2 authentication.

Level 3 - Send NTLM 2 response only. Clients use NTLM 2 authentication, and use NTLM 2 session security if the server supports it; domain controllers accept LM, NTLM, and NTLM 2 authentication.

Level 4 - Domain controllers refuse LM responses. Clients use NTLM authentication, and use NTLM 2 session security if the server supports it; domain controllers refuse LM authentication (that is, they accept NTLM and NTLM 2).

Level 5 - Domain controllers refuse LM and NTLM responses (accept only NTLM 2). Clients use NTLM 2 authentication, use NTLM 2 session security if the server supports it; domain controllers refuse NTLM and LM authentication (they accept only NTLM 2).

Method 3: Use a Password That Is at Least 15 Characters Long

The simplest way to prevent Windows from storing an LM hash of your password is to use a password that is at least 15 characters long. In this case, Windows stores an LM hash value that cannot be used to authenticate the user.

To get an idea of the power of using NTLM for your hash algorithm lets see how long it will take to generate a NTLM mixed-alphanumeric rainbow table with Winrtgen:

Rainbow Table properties	×
	in Count N° of tables
Charset	
mixalpha-numeric	Edit
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN0PQRSTUVWXYZ012345678	89
Table properties	
Key space: 3579345993194 keys	
Disk space: 610.35 MB	
Success probability: 0.043235 (4.32%)	
Benchmark	
Hash speed: 1721763 hash/sec	
Step speed: 723588 step/sec	
Table precomputation time: 2.55926 days	
Total precomputation time: 2.55926 days	
Max cryptanalysis time: 11.056 seconds	
Benchmark	OK Cancel

For a mixed-alphanumeric NTLM table it will take 2.5 days to generate one table with a 4.32% success rate.

Learn Security Online, Inc. ©	https://www.learnsecurityonline.com/	
Learn Security Online		
Rainbow Table properties	×	
Hash Min Len Max Len Index Chain I I 7 0 4000 Charset	en Chain Count N° of tables	
mixalpha-numeric	✓ Edit	
abcdefghijkImnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVW>	YZ0123456789	
Table properties Key space: 3579345993194 keys Disk space: 59.60 GB (610.35 MB each table) Success probability: 0.987963 (98.80%)		
Benchmark		
Hash speed: 1721763 hash/sec Step speed: 734214 step/sec		
Table precomputation time: 2.52222 days		
Total precomputation time: 252.222 days Max cryptanalysis time: 18.16 minutes		
Benchmark	OK Cancel	

It will take over 100 rainbow tables, 60 GB of space, and 252 days to create the tables to crack the same passwords (with a 98.80% success rate) we have been attacking throughout this paper if they were stored as NTLM instead of LM!

Conclusion

As you can see Rainbow Tables and RainbowCrack are powerful password auditing tools. The best course of action to protect yourself is to not allow the storage and use of LAN Manager (LM) passwords on your network if you don't absolutely need to and create and enforce a strong password policy that will force the storage and use of passwords as NTLM and not LM. Additionally, the time to compute and space requirements of complex Rainbow Tables should limit the use of them to only determined attackers or auditors. A strong password policy, strong domain security policy, and keeping up with your patches and updates is your best safeguard against password attacks.

References

RainbowCrack--Not a New Street Drug http://redmondmag.com/columns/article.asp?EditorialsID=736

Rainbow Tables: Nature, Use, and Generation http://security.the-engine.org/documents/48/rainbow-tables-nature-use-and-generation

"Faster Cryptanalytic_time – memory trade off" paper by Philippe Oechslin http://lasecwww.epfl.ch/php_code/publications/search.php?ref=Oech03



https://www.learnsecurityonline.com/

Project RainbowCrack http://www.antsight.com/zsl/rainbowcrack/

Get Free Rainbow Tables via torrent files <u>http://rainbowtables.shmoo.com/</u>

The Tactical Use of RainbowCrack to Exploit Windows Authentication in a Hybrid Physical-Electronic Attack by Mike Mahurin http://www.giac.org/practicals/GCIH/Mike Mahurin GCIH.pdf

Password Cracking: Rainbow Tables Explained https://www.isc2.org/cgi-bin/content.cgi?page=738

How to prevent Windows from storing a LAN manager hash of your password in Active Directory and local SAM databases http://support.microsoft.com/default.aspx?scid=KB;EN-US;q299656&

How to disable LM authentication on Windows NT http://support.microsoft.com/default.aspx?scid=kb;EN-US;q147706

How to enable NTLM 2 authentication http://support.microsoft.com/default.aspx?scid=kb;EN-US;q239869

Microsoft Windows 2000 Security Hardening Guide http://www.microsoft.com/technet/security/prodtech/windows2000/win2khg/03osinstl.mspx

RainbowTable Wikipedia Entry http://en.wikipedia.org/wiki/Rainbow_table

LanManger Wikipedia Entry http://en.wikipedia.org/wiki/LM_hash

About the Author

Chris Gates, CISSP serves as the operations manager and course mentor for <u>LearnSecurityOnline.com</u>. Feel free to email comments and suggestions on the tutorial to chris [at] learnsecurityonline [dot] com.