NAME – MANAV AHLAWAT
NET ID – ma5169
ISP Assignment 2

1) Formspring:

The Formspring's list of password hashes involved Secure Hash Algorithm technique. The passwords encrypted in the SHA-256 format had a salt value ranging from 00 - 99. Salt is a randomly generated number added to the passwords before the passwords are converted into the hash value and because SHA-256 is irreversible, it is very difficult to crack the passwords, even if the salt value is known.
I used Hashcat tool for cracking the passwords using dictionary mode and brute force method. I got around 243 passwords recovered using dictionary mode and around 20k using brute force for around 2-3 hours.

Ashley Madison:
Ashley Madison list of password hashes was encrypted using a cryptographic function called bcrypt.

Bcrypt is more computationally intensive than some other functions like MD5. Also the Ashley Madison developers used a cost factor of 12 in their implementation, meaning that each possible password an attacker wants to test needs to be put through 4,096 rounds of hashing. This makes cracking passwords with a small size dictionary very slow. Better idea to improve chances of password cracking would be to use a large dictionary size.

2). After searching online a lot, I first tried using John The Ripper, password cracking tool. But I could not get enough good results using John The Ripper(JTR). I even installed Virtual Box on my system and then mounted Kali Linux on it to use the pre-installed resources it has for password cracking. But it turned out to be very slow and I could not get any results. Also tried to use Hydra. But I found Hashcat to one of the better password cracking tools. The reason for that is Hashcat uses wide variety of algorithms to crack passwords and more importantly is uses even the GPU which makes it comparatively faster than other password cracking tools.

The two techniques I used for cracking passwords are-
Dictionary mode
Brute Force

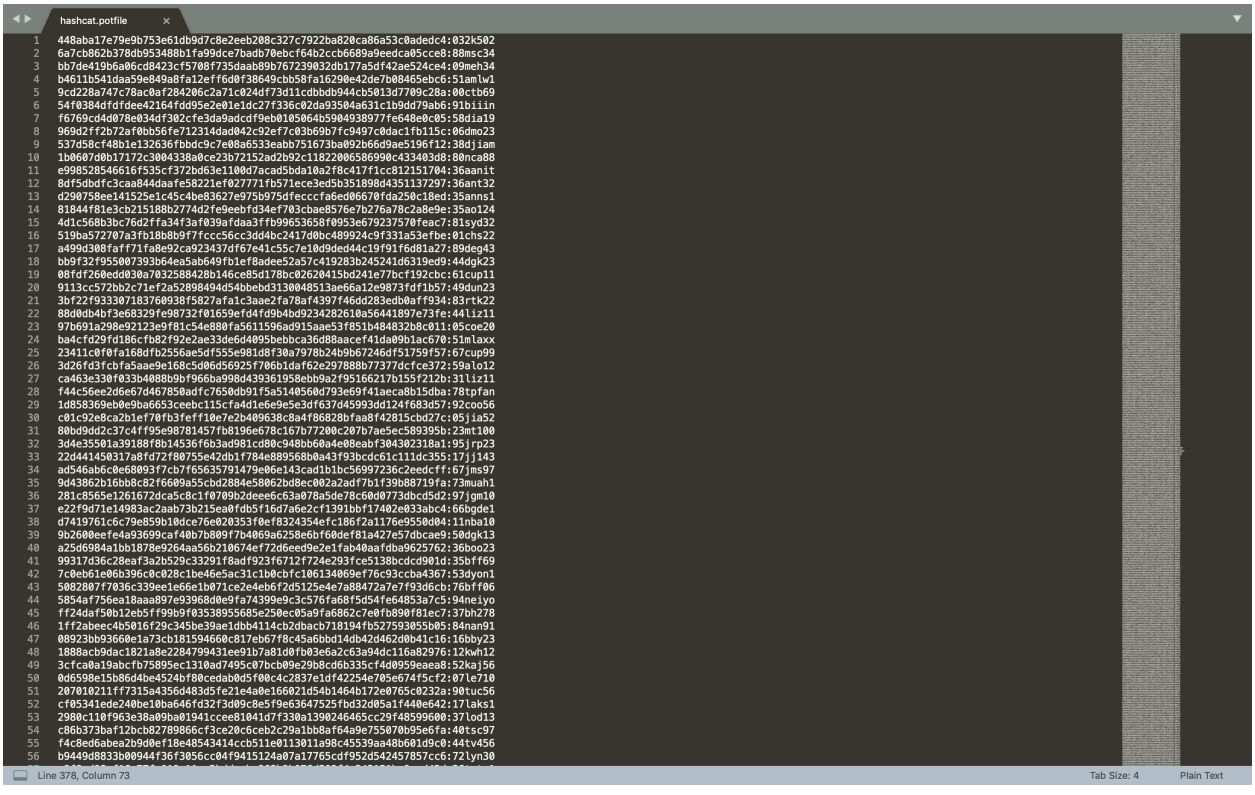
Dictionary mode attack is the simplest and fastest password cracking attack. The dictionary file has a lot of random words or previously cracked passwords which means we can hash them again using the same hashing technique which has been used for the passwords we are trying to recover. Now if the hashed values of the dictionary file and already hashed passwords

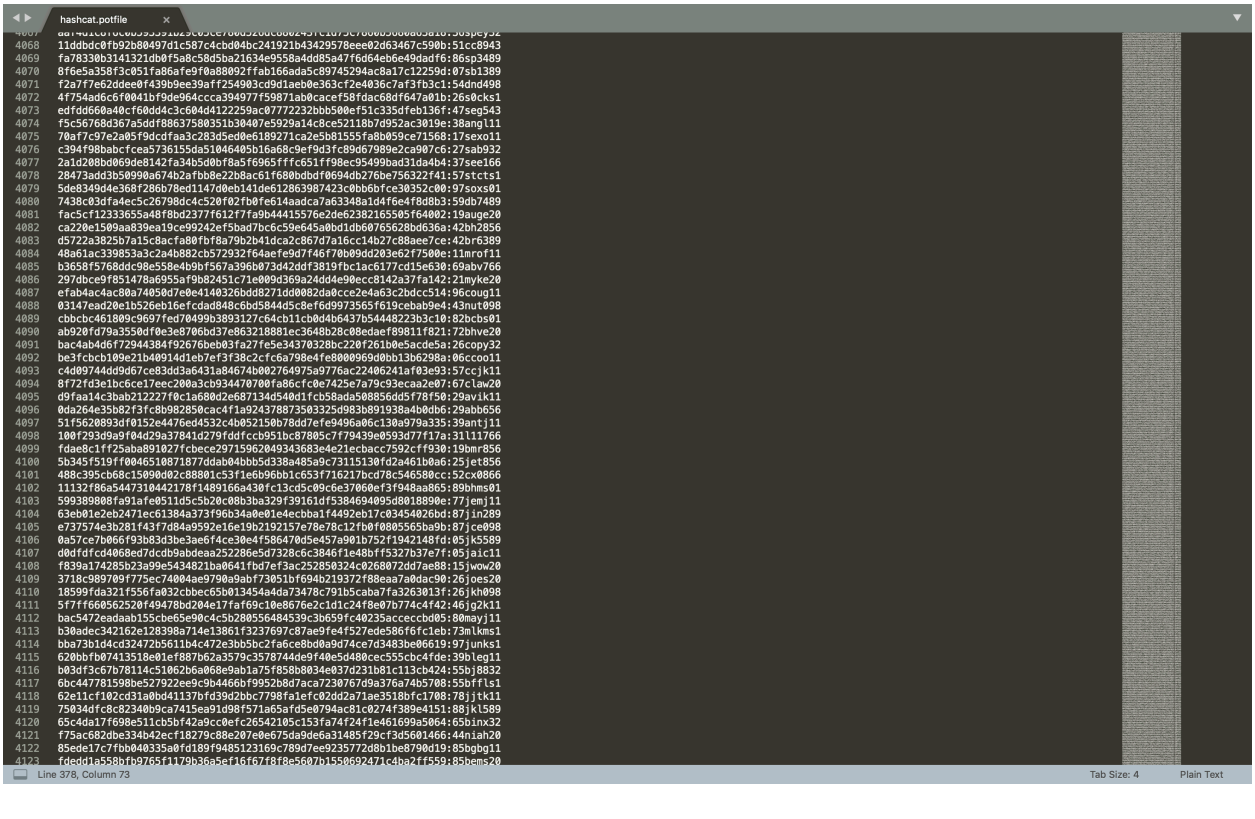
match, the tool displays out the plain text corresponding to that matched hash value. This attack is plausible only due to high performance capabilities of computers these days.

Brute force is one of the last options to crack passwords as it is really time consuming. Brute force requires a lot of computational power as it tries to guess all the possible combinations of all letters, numbers and special characters to crack a password. Advantage of using this technique is that if kept running for good amount of time, it can fetch many more results of cracked passwords than any other technique.

3)When I applied dictionary attack using Hashcat on Formspring's hashed passwords, I managed to crack around 4300 passwords. I used a common dictionary file which is rockyou.txt available on github. After getting these results, I added theis cracked password list to the dictionary file which I have. I again tried to crack more passwords and resulted in cracking more than 40K this time. I realized that using a previously password cracked list really increased the efficiency of cracking the remaining passwords.
Dictionary attack in Hashcat was done by using the following command-
./hashcat -m 1400 -a 0 formspring.txt rockyou.txt

```
Last login: Tue Feb 19 10:06:14 on ttys000
Restored session: Tue Feb 19 15:59:24 EST 2019
Manavs-MacBook-Pro:hashcat manavahlawat$ ./hashcat -m 1400 -a 0 formspring.txt rockyou.txt
hashcat (v5.1.0-42-g471a8ccc) starting...

OpenCL Platform #1: Apple
========================
* Device #1: Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz, skipped.
* Device #2: Intel(R) Iris(TM) Plus Graphics 655, 384/1536 MB allocatable, 48MCU

Hashes: 419564 digests; 419564 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

ATTENTION! Pure (unoptimized) OpenCL kernels selected.
This enables cracking passwords and salts > length 32 but for the price of drastically reduced performance.
If you want to switch to optimized OpenCL kernels, append -O to your commandline.

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

INFO: Removed 4206 hashes found in potfile.

Dictionary cache hit:
* Filename..: rockyou.txt
* Passwords.: 14344384
* Bytes.....: 139921497
* Keyspace..: 14344384

Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Type........: SHA2-256
Hash.Target......: formspring.txt
Time.Started.....: Tue Feb 19 16:00:31 2019 (4 secs)
Time.Estimated...: Tue Feb 19 16:00:35 2019 (0 secs)
Guess.Base.......: File (rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#2.........:  3680.2 kH/s (10.96ms) @ Accel:64 Loops:1 Thr:64 Vec:1
Recovered........: 4206/419564 (1.00%) Digests, 0/1 (0.00%) Salts
Recovered/Time...: CUR:N/A,N/A,N/A AVG:0,0,0 (Min,Hour,Day)
Progress.........: 14344384/14344384 (100.00%)
Rejected.........: 0/14344384 (0.00%)
Restore.Point....: 14344384/14344384 (100.00%)
Restore.Sub.#2...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#2....: $HEX[30323133446f6d] -> $HEX[042a0337c2a156616d6f732103]

Started: Tue Feb 19 16:00:28 2019
Stopped: Tue Feb 19 16:00:37 2019
Manavs-MacBook-Pro:hashcat manavahlawat$
```


hashcat.potfile
448aba17e79e9b753e61db9d7c8e2eeb208c327c7922ba820ca86a53c0adedc4:032k502
6a7cb862b378db953488b1fa99dce7badb70ebcf64b2ccb6689a9eedca05cce8:88msc34
bb7de419b6a06cd8423cf5708f735daab89b767239032db177a5df42ae524ce4:09meh34
b4611b541daa59e849a8fa12eff6d0f38649cbb58fa16290e42de7b08465ebc6:51amlw1
9cd228a747c78ac0af284206c2a71c024df73d11cdbbdb944cb5013d7709c28a:00ctb69
54f0384dfdfdee42164fdd95e2e01e1dc27f336c02da93504a631c1b9dd79ab6:91biiin
f6769cd4d078e034df302cfe3da9adcdf9eb0105064b59049398977fe648e0c05:58dia19
969d2ff2b72af0bb56fe712314dad042c92ef7c03b69b7fc9497c0dac1fb115c:06dmo23
537d58cf48b1e132636fbbdc9c7e08a6533eabb751673ba092b66d9ae5196f12:38djiam
1b0607d0b17172c3004338a0ce23b72152ad2b92c1182200658690c433403d8:80nca88
e9985285466616f535cf372bd63e1100d7acad5bda10a2f8c417f1cc812151704:36aanit
8df5dbdfc3caa844daafe58221ef027771fb571ece3ed5b351898d4351137297:36ant32
d290758ee141525e1c45c4be83627e975b975dfecccfa6ed06670fda250c18ed:35anns1
81844f81e3cb215188b2774d2fe9eebfd34ef703cbae8576e7b276a78c2a8e9e:35ao124
4d1c568b3bc76d2ffa34f3af039afdaa3ffb99653658f0953e679237570feac7:81syd32
519ba572707a3fb18b8b9f7fccc56cc3dd4bc2417d0bc489924c9f331a53efbe:01chs22
a499d308faff71fa8e92ca923437df67e41c55c7e10d9ded44c19f91f6d81a27:89deg43
bb9f32f955007393b64ea5ab649fb1ef8adee52a57c419283b245241d6319ed9:44dgk23
08fdf260edd030a7032588428b146ce85d178bc02620415bd241e77bcf192cbc:61cup11
9113cc572bb2c71ef2a52898494d54bbebd3130048513ae66a12e9873fdf1b57:49dun23
3bf22f933307183760938f5827afa1c3aae2fa78af4397f46dd283edb0aff934:83rtk22
88d0db4bf3e68329fe98732f01659efd4fd9b4bd9234282610a56441897e73fe:44liz11
97b691a298e92123e9f81c54e880fa5611596ad915aae53f851b484832b8c011:05coe20
ba4cfd29fd186cfb82f92e2ae33de6d4095bebbca36d88aacef41da09b1ac670:51mlaxx
23411c0f0fa168dfb2556ae5df555e981d8f30a7978b24b9b67246df51759f57:67cup99
3d26fd3fcbfa5aae9e168c5d06d56925f706b1daf62e297888b77377dcfce372:59alo12
ca463e330f033b4088b9bf966ba998d439361958ebb9a2f95166217b155f212b:31liz11
f44c56ee2d6e67d467850adfc7650db91f5a5140560d793e69f41aeca8b15dba:78tpfan
1d858369eb0e9ba6653ceebc115cfa4d1e6e9e5e3df637d45993dd124f683d57:92coo56
c01c92e8ca2b1ef70fb3feff10e7e2b409638c8a4f86828bfaa8f42815cbd27c:05jia52
80bd9dd2c37c4ff95e98781457fb8196e678c167b77200c207b7ae5ec589395b:23mt100
3d4e35501a39188f8b14536f6b3ad981cd80c948bb60a4e08eabf304302318a1:95jrp23
22d441450317a8fd72f80755e42db1f784e889568b0a43f93bcdc61c111dc355:17jj143
ad546ab6c0e68093f7cb7f656357914 79e06e143cad1b1bc56997236c2eedcff:67jms97
9d4386 2b16bb8c82f6609a55cbd2884e58062bd8ec002a2adf7b1f39b88719fa:73muah1
281c8565e126172dca5c8c1f0709b2deee6c63a078a5de78c60d073dbcd5d2:97jgm10
e22f9d71e14983ac2aab73b215ea0fdb5f16d7a6e2cf1391bbf17402e033abc4:66bgde1
d7419761c6c79e859b10dce76e020353f0ef8324354efc186f2a1176e9550d04:11nba10
9b2600eefe4a93699caf40b7b809f7b4069a6258e6bf60def81a427e57dbcae9:50dgk13
a25d6984a1bb1878e9264aa56b210674ef72d6eed9e2e1fab40aafdba9625762:36boo23
99317d36c28eaf3a2b529c33291f8adf923f6712f724e293fce5138bcdcd901d:35bff69
7c0eb61e06b396c0c028c1be46e5ac31c1b0cbfc106134069ef76c93ccba4367:53dyon1
5082807f7036c339ee1e66e1b071ce2e4eb6f2d5125e4e7a88472a7e7f93d6cb:76bff06
5854af756ea18aaa897e93968d0e9fa74399e9c3c576fa68f5d54fe64853a7c5:94neiyo
ff24daf50b12eb5ff99b9f03538955685e250ec05a9fa6862c7e0fb890f81ec7:37bh278
1ff2abeec4b5016f29c345be39ae1dbb4114cb2dbacb718194fb527593055b05:84nan91
08923bb93660e1a73cb181594660c817eb67f8c45a6bbd14db42d462d0b41c16:16bby23
1888acb9dac1821a8e2284799431ee91b7a81d0fb03e6a2c63a94dc116a82976:12kwh12
3cfca0a19abcfb75895ec1310ad7495c07bcb09e29b8cd6b335cf4d0959eaaea8:52kaj56
0d6598e15b86d40e4524bf80eeda0d5f00c4c2837e1df42254e705e674f5cf2:071e710
207010211ff7315a4356d483d5fe21e4a0e166021d54b1464b172e0765c0232a:90tuc56
cf05341ede240be10ba646fd32f3d09c8e5f9e63647525fbd32d05a1f440e642:17laks1
2980c110f963e38a09ba01941ccee81041d7f330a1390246465cc29f48599600:37lod13
c86b373baf12bcb82789866cf3ce20c6ceb2c29a1bb8af64a9e755070b95e6fa:40tsc97
f4c8ed6abea2b9d0ef18e48543414ccb511e0113011a98c45539aa48b601d9c0:44tv456
b9449d8833b00944f36f3056cc04f9415124a07a17765cdf952d542457857cc6:72lyn30
Line 378, Column 73
Tab Size: 4
Plain Text


hashcat.potfile
11ddbdc0fb92b80497d1c587c4cbd04bc241921b43429578eee02d63467c590b:51cc8943
fa78330b3141321db0f5a8c58d5ba21634e9558a4dd85a47f6d64eb6e49dadf6:86sp3489
8f6e5a358f3c051fa86afe9f0a88092ffab166ada5c89745294ac8a17c12259d:07sb1389
f2a7f7e62ddee0f439b9ee39aff254903cd9d1aeb0e363cf2e4036c7af3fb3c1:54dnd498
4f754ad6c6f0041bf9de964ccca394977f59871ab0cacef58fdac68df647a01b:26s0cks1
edfdd660a40cf60dd4c3c604d4122259ac07772232bbb500ef51c335dfeb136f:47seg543
f5c56768d367a5ddf80637500351b30407e5929a14c8ce52110b7d952ac3019e:38angl11
70af7c97e2a05f9dcdfaa3c283d5ed0e6189271ca2e5b81555fa8b059ce71568:17sexo11
c394f98babcfcea5736155da51046405b16a68c9ef9d3fc89d67989e2ca907c6:27sab932
2a1d208bd069de8142fa34b5d0bf8a5f6965fffc651ff98ec95499bad31da4b4:37see166
28473add3b50990a674b2afbb8e22b8ac61f680bdbdf0694db276be756322f41:57stcts1
5de8349d4e368f286b78ed1147d0eb141de612863987423c0bb6bfce30352c00:97soxs01
7438c03dfa4ec5c26798dc4c520f02fb0fe614aadca7a63348a1d4f6e4f8d8b3:59db7489
fac5cf12333655a48f8bd2377f612f7fa9b44415576e2de623821655 05f64002:19auge20
ca220e1509aa839ea19ce99242ef5bad7bc6c59e645a0bd1db60765628bd636a:42bh2856
d5722a3825b7a15c8acfa80fbf8a79b2b41dca2c867d7a16cc14b27c88aee7ce:42br6389
48a61ac339853a3c2a4b882cb572932f64aefe9d7f46f70b09dd203e62f7500e:11mrof11
b3658f5768ddc98e558e4b9bf567a396b073d42ddf3819fbc1ac6177cd15e630:69abv766
297dbce9f851478a6955af9b82451c71e009d369a24dd4e90ecc8142a37fa142:21myke20
efab4ac4ac80a74050d7e0e4140326bdd8271dd0082da0cce2e4a63c2bdcc514:96coug11
03147ead20e1b526eb16ef0dad848c6bd5844d5c8ef6d9973565f619cebab9e4:43nut090
cbbcbc461809c9697fed70490b389317 7c64cb11cb0d4b6437b54448223b3c10:82bcbs01
ab920fd79a3550df0e3e8706bd37e863218881ec364Bb28c08edaef89811f821:72bhve20
bac4ab4d6f72944384f92676beb03fa27fe5e34370328bc220f261b0e5acde5a:26copy32
be3fcbcb109e21b40914d1eb7ef3f38c2cfc6a798e4fe8000969d0bb13b6298a:06ccoo11
c4d09744dd9d67ce83dd3a6431a84674b002764975a9776ac22406241af03e93:57ccjk11
8f72fd3e1bc6ce17eec200a3cb934470700fa86cfc0e7425e7a79c93ecaa2e07:67claw20
d9faa14c3bab212227f0c1a680d2e687134d540d1fcb58ada508856dd5f78f50:29avik11
0da264e35b82f3fc8b982850cac4f1a924378873503325d983b891938a4b0db6:29abb656
51f5620893df0152e4476ed4532c4b052150b78dd7efe9496a06c130a979a335:39antj11
100f293d9a9f04d29a37841d279fddfccb951bc87805c7f79439e0593d77f17a:31l11766
fdae8c1f25aba891027fcbece29715962049343683e4e21ecbacc7592cff930:19bmr856
5b345f519ff0046510871877ddab04bbb5d338a485a9c73115130fd2a461b0a6:25jet856
488c395cb68c15090d02c88801c53f1e6b96bb1c653f716217bcd78c546580cc:52cx0866
11132f86a5447310442178f1489166a430fc13cd5c09c6e3706def3f948aaa865:89bhms01
599389088fa91afe0511d5c5b20c08b3ba329f39161df538469409 5d8018890e:77jemj11
63eb01e2eb2471ec61384a373f96b34a0e01e4ecbba1f4492eb17c03454026bd:87jin289
e737574e3b281f43f7d84a9592e16e19b2192157e78e78c12fb0f0805565ba9b:87jce098
0a57ce7b066f93b83d3be3ae6f4ce30e4f586470d5e457a901b752f1942148fd:05jk5589
d0dfdfcd4068ed7dcdb9abdeaa252286e5d7328c6c3846f1e48bff5327b37e7f:05jaic11
f839a174285b23a99e5434821ba0641fbd7ef3ac252850324c0268072dd7ae88:15jwow20
3718c989709f775ec74004ae9790a9abf73051bf694b219372f88eaa7a0dac90:26joes20
18599fda321f556fa032cbbec65b013454a4da73478c791b5caba7fa3263013f:26jkh098
5f7ff660562520f49478bd204e17faf69c10e8676e2c1d1c24f8e07b774c4f42:46jg2k11
bac5472eadaab155cbe65e90c4c5b280556484539c3b659fc40235accccbd05:80mayj11
b30adec342162e128398a714e13861f3237697c87ae9fe4f527ede586f6fc1eb:73mlkms1
bba73b1d4cd32472b5611b4c472e3bb5bc2fa6ce8bd0a9d74ce7d3483be06619:97junks1
620bbfb07413518e01ef887b62a3579c3578744de9f40e5d480cec555cbc4fde:39bleg11
b03df3c67b78114c51062b6a068e9ab1a75f858b8034e037d231b81c113cb424:55bj8832
6bc447781598be527922ee9c9b4466bf9052b8d7aca7230767d9a376a74b5708:55bffls1
62e11cf102cd31a0bd41137bfd39d2bbc7798fa1efc02dd2a71ae3518bfc1708:78jjtk11
75034dfc8c82340b9ca7415ea91d98f571a65afd5e0794ac81c0274f389e4151:88jkl589
65c4da17f698e511cb5bf42a9cc0efc26542105c153fa74f24f1e461699a5064:65binx32
f75ac682dbe334b42ecf18279c88e207e0e675da9de6a31485f29cf3d5601466:44niah20
85ede17c7fbb040335a0fd189f9485123169c789d7ee9235772d8b1be8790d19:85bgbg11
fdedd1a558bfb9765f1179b36a5ef16f67f8f8e5607b1550692471c4ba2ff559:85bems20
Line 378, Column 73
Tab Size: 4
Plain Text

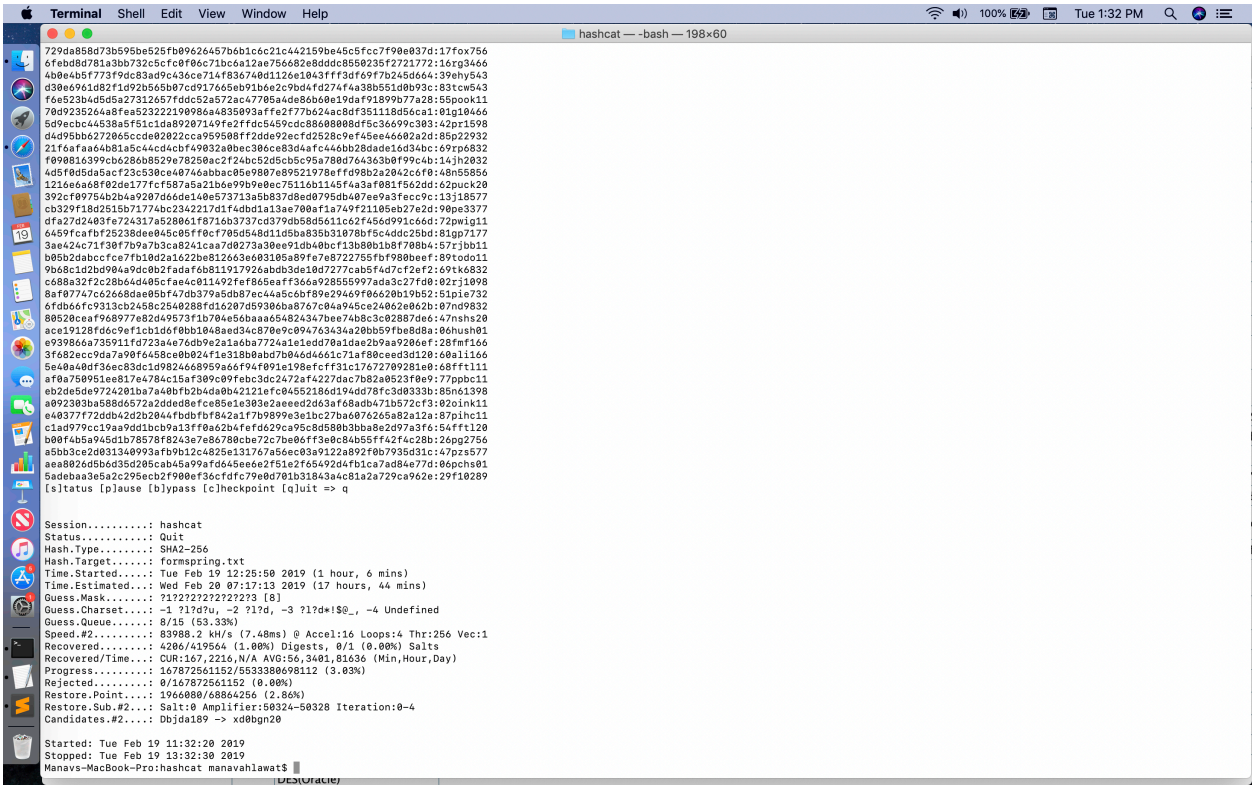
Then I tried to do brute force attack, and although it was really slow, but I managed to crack a lot more passwords than with the dictionary attack. The command used was-
./hashcat -m 1400 -a 3 formspring.txt

Cracking passwords for Ashley Madison turned out to be really difficult. I believe this was because they had a pretty good hashing technique which has bcrypt. After running the dictionary attack for around an hour, I managed to get only two passwords cracked. I had to stop the attack as the stats showed me that it would take more than an year to crack all the passwords.The command used was –

./hashcat -m 3200 -a 0 ashley.txt rockyou.txt –force

Using Brute force on Ashley Madison did not fetch me any results, as it was very slow and my system started to hang. I ran brute force attack for more than 3 hours but could not get any passwords cracked so I just aborted the operation.
The command used was-

./hashcat -m 3200 -a 3 ashley.txt –force




```
729da858d73b595be525fb09626457b6b1c6c21c442159be45c5fcc7f90e037d:17fox756
6febd8d781a3bb732c5cfc0f06c71bc6a12ae756682e8dddc8550235f2721772:16rg3466
4b0e4b5f773f9dc83ad9c436ce714f83674d0116e1043fff3df69f7b245d664:39ehy543
d30e6961d82f1d92b565b07cd917665eb91b6e2c9bd4fd274f4a38b551d0b93c:83tcw543
f6e523b4d5d5a27312657fddc52a572ac47705a4de86b60e19daf91899b77a28:55pook11
70d9235264a8fea523222190986a4835093affe2f77b624ac8df351118d56ca1:01g10466
5d9ecbc44538a5f51c1da89207149fe2ffdc5459cdc88608008df5c36699c303:42pr1598
d4d95bb6272065ccde02022cca959508ff2dde92ecfd2528c9ef45ee46602a2d:85p22932
21f6afaa64b81a5c44cd4cbf49032a0bec306ce83d4afc446bb28dade16d34bc:69rp6832
f0908163399cb628b0529e78250ac2f24bc52d5cb5c95a700d764363b0f99c4b:14jh2032
4d5f0d5da5acf23c530ce40746abbac05e9807e8952197beffdf98b2a2042c6f0:48n55856
1216e6a68f02de177fcf587a5a21b6e99b9e0ec75116b1145f4a3af081f562dd:62puck20
392cf09754b2b4a9207d66de140e573713a5b837d8ed0795db407ee9a3fecc9c:13j18577
cb329f18d2515b71774bc2342217d1f4dbd1a13ae700af1a749f21105eb27e2d:90pe3377
dfa27d2403fe724317a528061f8716b3737cd379db58d5611c62f456d991c66d:72pwig11
6459fcafbf25238dee045c05ff0cf705d548d11d5ba835b31078bf5c4ddc25bd:81gp7177
3ae424c71f30f7b9a7b3ca8241caa7d0273a30ee91db40bcf13b80b1b8f708b4:57rjbb11
b05b2dabccfce7fb10d2a1622be812663e603105a89fe7e8722755fbf980beef:89todo11
9b68c1d2bd904a9dc0b2fadaf6b811917926abdb3de10d7277cab5f4d7cf2ef2:69tk6832
c688a32f2c28b64d405cfae4c011492fef865eaff366a928555997ada3c27fd0:02rj1098
8af07747c62668dae05bf47db379a5db87ec44a5c6bf89e29469f06620b19b52:51pie732
6fdb66fc9313cb2458c2540288fd16207d59306ba8767c04a945ce24062e062b:07nd9832
80520ceaf968977e82d49573f1b704e56baaa654824347bee74b8c3c02887de6:47nshs20
ace19128fd6c9ef1cb1d6f0bb1048aed34c870e9c094763434a20bb59fbe8d8a:06hush01
e939866a735911fd23a4e76db9e2a1a6ba7724a1e1edd70a1dae2b9aa920bef:28fmf166
3f682ecc9da7a90f6458ce0b02a4fa318b0abd7b046d4661c71af80ceed3d120:60ali166
5e40a40df36ec83dc1d9824668959a66f94f091e198efcff31c17672709281e0:68fftl11
af0a750951ee817e4784c15af309c09febc3dc2472af4227dac7b82a0523f0e9:77ppbc11
eb2de5de9724201ba7a40bfb2b4da0b42121efc04552186d194dd78fc3d0333b:85n61398
a092303ba588d6572a2dded8efce85e1e303e2aeeed2d63af68adb471b572cf3:02oink11
e40377f72ddb42d2b2044fbdbfbf842a1f7b9899e3e1bc27ba6076265a82a12a:87pihc11
c1ad979cc19aa9dd1bcb9a13ff0a62b4fefd629ca95c8d580b3bba8e2d97a3f6:54fft120
b00f4b5a945d1b78578f8243e7e86780cbe72c7be06ff3e0c84b55ff42f4c28b:26pg2756
a5bb3ce2d031340993afb9b12c4825e131767a56ec03a9122a892f0b7935d31c:47pzs577
aea8026d5b6d35d205cab45a99afd645ee6e2f51e2f65492d4fb1ca7ad84e77d:06pchs01
5adebaa3e5a2c295ecb2f900ef36cfdfc79e0d701b31843a4c81a2a729ca962e:29f10289
[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => q


Session..........: hashcat
Status...........: Quit
Hash.Type........: SHA2-256
Hash.Target......: formspring.txt
Time.Started.....: Tue Feb 19 12:25:50 2019 (1 hour, 6 mins)
Time.Estimated...: Wed Feb 20 07:17:13 2019 (17 hours, 44 mins)
Guess.Mask.......: ?1?2?2?2?2?2?3 [8]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 8/15 (53.33%)
Speed.#2.........: 83988.2 kH/s (7.48ms) @ Accel:16 Loops:4 Thr:256 Vec:1
Recovered........: 4206/419564 (1.00%) Digests, 0/1 (0.00%) Salts
Recovered/Time...: CUR:167,2216,N/A AVG:56,3401,81636 (Min,Hour,Day)
Progress.........: 167872561152/5533380698112 (3.03%)
Rejected.........: 0/167872561152 (0.00%)
Restore.Point....: 1966080/68864256 (2.86%)
Restore.Sub.#2...: Salt:0 Amplifier:50324-50328 Iteration:0-4
Candidates.#2....: Dbjda189 -> xd0bgn20

Started: Tue Feb 19 11:32:20 2019
Stopped: Tue Feb 19 13:32:30 2019
Manavs-MacBook-Pro:hashcat manavahlawat$
```

```
Started: Tue Feb 19 16:02:03 2019
Stopped: Tue Feb 19 16:02:26 2019
Manavs-MacBook-Pro:hashcat manavahlawat$ ./hashcat -m 3200 -a 3 ashley.txt --force
hashcat (v5.1.0-42-g471a8ccc) starting...

OpenCL Platform #1: Apple
========================
* Device #1: Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz, skipped.
* Device #2: Intel(R) Iris(TM) Plus Graphics 655, 384/1536 MB allocatable, 48MCU

Bitmap table overflowed at 18 bits.
This typically happens with too many hashes and reduces your performance.
You can increase the bitmap table size with --bitmap-max, but
this creates a trade-off between L2-cache and bitmap efficiency.
It is therefore not guaranteed to restore full performance.

Hashes: 2200290 digests; 2200290 unique digests, 2200290 unique salts
Bitmaps: 18 bits, 262144 entries, 0x0003ffff mask, 1048576 bytes, 5/13 rotates

Applicable optimizers:
* Zero-Byte
* Brute-Force

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 72

Watchdog: Hardware monitoring interface not found on your system.
Watchdog: Temperature abort trigger disabled.

INFO: Removed 2 hashes found in potfile.

The wordlist or mask that you are using is too small.
This means that hashcat cannot use the full parallel power of your device(s).
Unless you supply more work, your cracking speed will drop.
For tips on supplying more work, see: https://hashcat.net/faq/morework

Approaching final keyspace - workload adjusted.

[s]tatus [p]ause [b]ypass [c]heckpoint [q]uit => q


Session..........: hashcat
Status...........: Quit
Hash.Type........: bcrypt $2*$, Blowfish (Unix)
Hash.Target......: ashley.txt
Time.Started.....: Tue Feb 19 16:03:56 2019 (1 hour, 48 mins)
Time.Estimated...: Wed Dec  5 11:39:07 2085 (66 years, 289 days)
Guess.Mask.......: ?1 [1]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 1/15 (6.67%)
Speed.#2.........:        0 H/s (2.94ms) @ Accel:1 Loops:1 Thr:8 Vec:1
Recovered........: 2/2200290 (0.00%) Digests, 2/2200290 (0.00%) Salts
Recovered/Time...: CUR:0,0,N/A AVG:0,0,0 (Min,Hour,Day)
Progress.........: 421/136417980 (0.00%)
Rejected.........: 0/421 (0.00%)
Restore.Point....: 0/1 (0.00%)
Restore.Sub.#2...: Salt:6 Amplifier:49-50 Iteration:2604-2605
Candidates.#2....: H -> H

Started: Tue Feb 19 16:03:39 2019
Stopped: Tue Feb 19 17:52:32 2019
Manavs-MacBook-Pro:hashcat manavahlawat$
```

Estimated time was more than 66 years.

Also, there was a certain pattern in the cracked passwords. All of them were a combination of numbers and characters. Almost every password started and ended with numbers and had some characters in between.

There was a lot of difference in speeds of both the attacks. Dictionary attack was kind of fast. On the other side, brute force attack took like forever to fetch me results of cracked passwords.

4)Based on your experience, describe (briefly – you can use a bulleted list for this one) the steps you would follow to try to crack passwords in the future. Basically, if you are given a hash file (one with just a bunch of hex values) for company X, what would you do to try to recover as many passwords as possible.

->First, it is important to identify the hashing technique which has been used to hash the passwords because only then can we think of an approach to crack those hashed values.

->The second thing would be to write a piece of code for cracking those hash values or using a suitable password cracking tool(I used Hashcat above).

->Because dictionary attack seems to be the most efficient way of cracking hash values, third step would be to prepare a large relevant dictionary file which could help recover as many passwords as possible.
Next, it would help if one could recongnize the salts that have been used to hash the plain text. It would really make it simpler to crack the passwords now.
Next step would be to use the correct and most efficient attack modes using the password cracking tool (Hashcat in my case).
Once, you have a initial list of cracked passwords, the best method to improve the efficiency is to use the already cracked password list in the dictionary file and then use the attack mode again to get better results.
If the above steps do not get you success, one can always try the more time consuming brute force attack.

5) Finally, if you were in charge of creating a password scheme, describe at a high level what algorithms and storage methods you would use. This is an open-ended question, so provide as many details as needed to justify your response.

If I was in charge of creating a password scheme, I would either use a hashing technique from SHA family and then use salts with it to make it stronger and more difficult to crack. But the drawback for this technique would be that, if someone is familiar with the salt that has been used in the hash value, it becomes relatively easier to crack such hashed values.
So, from my experience after working on this assignment, I would go for bcrypt encryption technique to hash the plain text. Bcrypt seems to be the strongest hashing technique which was used by Ashley Madison's developers. As a result, I could not get enough passwords cracked in a limited amount of time.