Xtreme In Security

Problem Statement

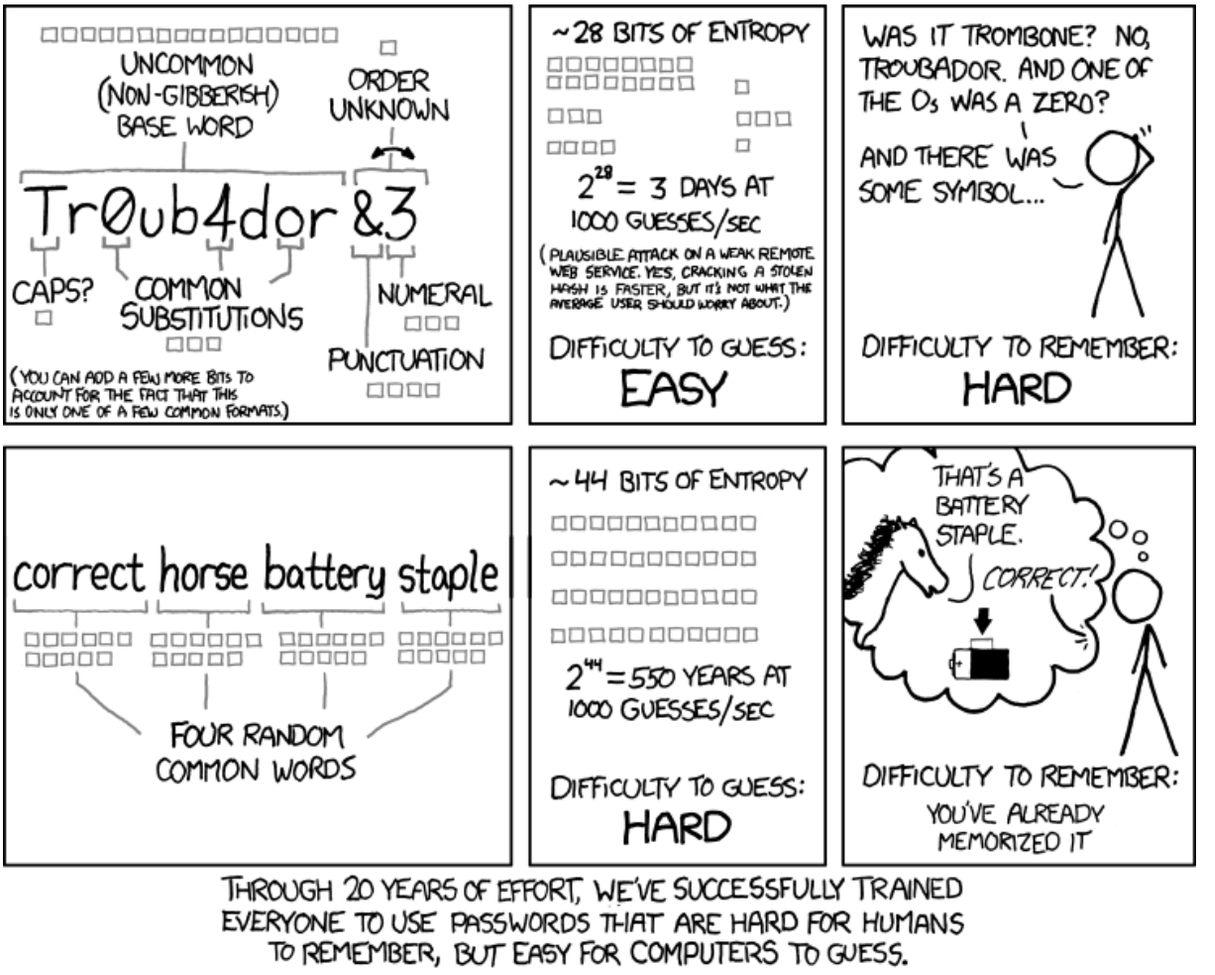
The Xtreme In Security Corp. has devised a password-based authentication system for their new operating system. They have made some unfortunate design decisions that make their system vulnerable to attack.

The system stores in a file a salted and peppered password hash. All of the passwords use the same salt "IEEE" and the same pepper "Xtreme". When the user is enrolled, they supply a password *p*. The system checks to make sure that *p* is less than 20 characters long, and that it contains only lowercase letters and numbers. Then the salt is prepended to the password and the pepper value is appended. The resulting value, using UTF-8 encoding, is then put through the SHA-256 hash algorithm, and then base64 encoded. This base64 encoded value is stored in a password file, along with the user's name.

Note: Even though the password contains only lowercase letters and numbers, the salt and the pepper contain some uppercase letters. The SHA-256 hash algorithm is case sensitive.

When the user is authenticated, a similar process occurs. The user is prompted for a username and a password. Then the salt is prepended to the supplied password and the pepper is appended. The resulting value is hashed with SHA-256, and base64 encoded. If the resulting value matches what is stored in the password file for the specified user, then the authentication is successful.

Your task is to break as many of the hashed passwords as possible to try to convince the system designers to improve their approach. Here is some inspiration from XKCD:



Credits: XKCD

Input Format

The input will consist of a single salted, peppered password hash. This hash will be derived from a single password.

The hash will be chosen from the following set of values:

```
/PtjJboZGlsmTovvyOhBOoTVnQKUP/gJXxjLAW9Lppw=
05HwH93tksb69U1ifesCQuYFP+gKPVH2L6W8JeBdXy0=
0BkyqI3NHyjh0m20wNt6txW08dgISMP4/qzUEezq4Aw=
1mT5cdKRz4BbfMdc8LAdnxfjsGO4lV0k0/V1IHtidmY=
28AdfW0JHmCP4TbieGON8dafRaFpUgpzuX2bHZN6WsM=
3hoie8omUyvM/9Qfx9dKfoptlwemYe2os8aohTGzoyw=
3uDaglldYUn11AadEhELBjE15A0L6hAaWnZmjCGtt+M=
4D4NstlYSjVN826Q+SXUDDmXgIJplpYWiJYf9rt7H8U=
7FCsEXCDTAxyLh3EPnNx7YrJ44SzehQYv3GmPSA6pWk=
81X3NN5JgTuGgqq3ErF0eL/l/wZFYkCwur6fZ06L2Us=
8FzGA/nS7XizLrAVOr/FoeKSq4gaoQRq+kpBKNXHlzc=
8OtpJ+E1XHv2RDsKlEwJc9KUFwPRzaqeHJ735Er6AVE=
8cdgZu9dBOrcTBMqElM+y9Vh5FTBRQ7n9EGa/4qVHUk=
8esDbw8ZVmUuUMEy2Scf5qGiZYiykevrvKpq2bVYHj4=
9DS4orbhPFbjJcosEqQg+eg0Si5qSOnftfHiqK8sYug=
9XDFlu4RPH0EL5XR+5VYILJ3UFAyltpfjONJKp/vcLk=
B2E/K/DywbLEKutOKpS8HxHFrZwucwac4KjzYgsXg3g=
BJQeqlV+4ejv0je5GpekzGdHHWHL5nnrbtD/170LZCA=
BjQiH/A2FUNHlUwhBi8NWmj3HmhmAh6Ag0kRyVSaVo8=
BwbAsOqPsxteVCpAwljrhYogsUS1bF/bLns2QdcLYUI=
CYDZabjeyTAwcEDEcvrX83514UmpjzvQUQ68DIZ/PXg=
CornANxoZ5FJnlhwHmK42CDXf3h6jFr3g73YIRuoymQ=
DDa2TJX20pPsNftfyJ3s6LBwSMSR3EADZGDxW2wThbs=
FFXy3vru2D8rTWZRlh9lSMvtEusfWgO17OmJCTQTECs=
FGkqFC/jLDqDZ10fqI1nGw7AQNWioOrZ3ydEaJyXBwo=
Fz+Y0H/R2rMZlc1C88Yx0A0xluYnVTinlw5qaSx8vWQ=
GcJMWMDF6+f+onf6oi1FbpnN7dVrFEZnlXtHqmaygs4=
GsXTQM0w+Clb1c9B7n28mADU2quLeI1n91KTyBboeHl=
HLnuqQmCYetzrau3frCDEpZ52QCIby108gugsmwAwQ0=
HWv9gx+GL/6g+0b0eOc+1Z/8BHse91/5T/DdiDwEknU=
IDB/pOthrWobzapJ/N8HsraNhwfbExAa2uusdiKHFFI=
IXYqlHbVeONERbxFe8SaEPEEKex45EihiC/l8CR52kk=
Idvs4Al9YZsqPG8xkSxVqb6MOVhbw5k+qtF8UZKYVE8=
IxQxcFXR51q8h8FLblPhYfUR30llAt6hX8TjZWVa/GE=
JCmqBN0MsW13tEmsQPYWg9Fj25MUrqFYvSK2arxTt0A=
JOXxLH/i8i+fxDlWP2cts5Si/5En1A8M3s/vy6Aadic=
KudA8vCEQdGaaCSxotpAcluXnVPS3MAZPkwl/lVupak=
LGZEfbUr/tMREpJtsao/uuewcJXApmgfHDbh1zzfdhU=
LZIzmWEqXDPJsnKttFGRaG/jUhHrbTEKt1XCO6XbdME=
Lq0kV5M0HDSgB4m5KZbbn6BYRNlkKfaaAr3/11ueopY=
Lqxt1UjT1ecV6ucgYn+yrGSUTxPWkZgdDtbygGwC/BA=
MHQTB1MSsvhBxMpdRUiaM9Uj1QxU7zYq3FqDlW2HT2s=
MKewBZryb2l36Y0tDyx+WuVeXUGfiSzcJplm9y1w9m0=
N5aunKGHeN2WETLXLzfhfCxAfkwtGU/imziiTF3t7oY=
NmrOUzHxKSfNT8UJ1YXbRL8I+HCAb+gIJ0bBXcHfagE=
NnSS+AuW1Z6zytSfqailVp4xxHHe70Av+IdhDlkoltQ=
O9L1ZWYwuzgaImTjOwuogXfpC+C44zzcDhpt08LjR5c=
R/ye+L9W+l6hyZ/v1POsWYboEGemlisARL8ohUvfBLI=
R1v9fEb9VrZuU5xiYTKTqhHF03VtXg7+KtfFHPkQuCQ=
RVfhsLovxa+/6tWgeSBASIlkzXkVtDPT4yYvjboHhlk=
Rz38Ng2qI3mPkaRB6uDoCYmmfzbVTCzpt2sG1o+TZqo=
S98FBzlv2vMVP/q+23m1wrHMJlrcy1rhoQGy338c4Bs=
SzraQWWasG5ZO1tJq16DqU/7M/o/WRiAWRl1aFFvwr8=
TInfNYwXvofBA+9QIe3+XEfDpO5ER+R+Jn3BOshhZWU=
VDkcRd54BhYlK5Wg2PPDa/jzGrSkMepGIv6Tw3I2ksc=
VjFTqTEY27V8lK2yCvhLhYm2Brh9bN1vWckVaevsiiY=
Wau4ReopjFKk8SYYNq5llBL+Rgg8aBR6h9UgTlv2u7I=
XtEWsXf16y6Bc7vQxDy7hwRdBVWo3dV9C6CDVSf4PLs=
Y5b6UztMueFYIFMl4a61jlD/ZhFG74/rVn8XaqqU+8c=
YollqBlewcxE/kF6PKvv0r1CLZkKx82657bB0eQbiK0=
YzSqlQTtq5j+Kd+hW1lSgBW0mn61vsQsxNeipq0sYCo=
aJlH+q0YjYZCpierKtbue5JDtZSF8tKxVKuHYUPQ65k=
bi6rh2HgTbJxR2GOTNWZLlxiiWZVnObptGj0KqOCSYo=
cX7VyMvSYhuRvEfAUb3uNh8kmjpNFg0tatUPN562iOg=
eSCTiCrzHPbngPu3F4ivPkLUv7MqLUlmWAhA4UO975Y=
eUqkcVCbgIx1bGhhmnN5MxJFJhVruINmG65TjT/EQ1k=
gE7PseB3mspPtYG3JROzT9FeqTfPFYQvBF2SJD9V19Y=
gMi4hC4o7Fmv6ylrU48BVy8I1khXwkaD36G7bWiZHeo=
h84yifAWGLj9sakEqxZ3QEjkXL42AoScP3L5Tdevm98=
```

```
hEGKCiTZSA5x560hodRoIBBTE8pv4sP1VXG4D0fXWcI=
ix+0IJlpLpHeSHEll+Q/IVY+FlRXn3xMA0ey6UlTi34=
j2GTUqtqZpotY8wF16zkvnbdCLTnX3oOZ30SjQUnIUk=
jtUg00EsmzzFkk8JgKg3OpkmPRpN9xbwsdNXQSPczwo=
k1J9Dv3El442CO6A2FGN1H8JgFO2kjNBvkjDR0WlvkA=
kuRpklh9kaNz6XvG8U6GO/IARH/SqhRnTiJbZHXC0yQ=
lFgqRrqTz1WXmO22u+ls1ZmWWUtuYrTJigsSB7I9NHI=
lUfxHX9xH2aOHheMMqQF+f5BNh97avew2uOwEN3B7HE=
m0IeuugWDOl9cFUFRYJhouCBT39T0dpp1xBOKPqHP94=
mgA5kgALstQpGUBp4vZ8oiz0P4jjAGl0wjgls6kQyMA=
nMAwaDYvElAwoqtqWMpBAWdhuRgRq/fmwWbRM7cOMlU=
opBtoC66YDRbLNqZAAu2FleKfF0HMOGHCOCPYeNHdx4=
ot/igM+me4e3UTT731qcBkSAcToyADMCddr7i7LCWGo=
qrkd/8imuRtiDb9N1w2hQRxJAkdx3Wqh1HVXPS7dym8=
r6BN0tdyAYZD0Nmc7bfV0WRcFBb1A2WlPPKHVRG59k8=
rdALvOYVhA3hnUTBlaQXigWBSgMYzGTreSKyMoAoKfw=
rjwtKqkPc7cfQ+zZ+E9c+fzQYhRvhVtaKEFb+srlHcQ=
rwvmTDiJxlEETbsngvpxYGwZZK+FGo7527odGuQUjtQ=
socJeO02bT2w+XZUrLoopbZvQ1lRhDfE88GVrJQ8p+g=
tDdmKQpMiVDFA1YdblkHSFzL4Z9UlQ9FSouf3TybOu0=
tojYiNtlWmq+7r1dSvxDk3W5at/NMAi1uxCHY61WAKk=
tqpGCBzhR+0ONFk1sBiHPhz+kRiXmY3CGdUXqnMJwLg=
uAZthS7b4ySZtWpM9pJ7ulYnhFdpFABpR2iPRQEmff0=
uCG9dSBejCOrZWsX7+u8G340p74s8lDS/El8MIeOyMo=
ugclIpDID0R1uFqBAcN3PNXhwlhen77GdAccFgpbs+A=
usg8BTtSfewL5M3OVg91TJCTc5vONLqgUCC/Si1Grsg=
vs2sCU8qG0pDYQfcjlPzDzvcbJnhP1OgFRcXP4i3ffw=
wEtqAs8JHjicWnXshAWF5Sg6NoswuG9qeJ7USw7YD7c=
y+zbMpySKY+WF97KkgRQ+tBpM7iTqTj9guWmGJcqfyA=
y1R6JQiUzUovgtdrvCkbeQAyMhFoupzhI5ZuQVPfCgw=
zqKPAOt5ziHSeRxc0TgUZF3rJxzBHAKdJeccvt3F7Jg=
```

Output Format

The output should consist of a single alphanumeric string, which hashes to the corresponding value given in the input.

Note: You may not be able to break all of the hashed passwords. If you are given a hash that you cannot break, you may output an arbitrary string. You will earn partial credit for this problem for each password that you do crack!

Sample Input

```
tDdmKQpMiVDFA1YdblkHSFzL4Z9UlQ9FSouf3TybOu0=
```

Sample Output

```
password1
```

Explanation

For all passwords, the salt value is "IEEE" and the pepper is "Xtreme". First, we can take the SHA-256 hash of the UTF-8 encoded string `IEEEpassword1Xtreme`. If we then base64 encode the result, the value equals the desired value: `tDdmKQpMiVDFA1YdblkHSFzL4Z9UIQ9FSouf3TybOu0=`.