

Forum:[Hack Forums](#)

Thread: [Cracking SHA-256 Hashes with Python \(Hashlib and IterTools\)](#)

Powa. - December 11, 2014, 5:53 pm

This tutorial is mainly meant to show you the basic of using the Hashlib module, but in a rather fun (although, almost useless) way.

We'll start by importing our modules and asking the user for the hash they are looking to crack.

Code `#!/usr/bin/env python`

```
import hashlib
import itertools
```

```
hash = raw_input("Hash: ")
Then we'll add in an alphabet for bruteforcing.
```

Code `#!/usr/bin/env python`

```
import hashlib

hash = raw_input("Hash: ")
alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
Then create our function for creating a SHA-256 Hash.
```

Code `#!/usr/bin/env python`

```
import hashlib

def create_SHA256(letter):
    hash_object = hashlib.sha256(letter)
    returnHash = hash_object.hexdigest()
    return returnHash

hash = raw_input("Hash: ")
alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
hashlib.sha256(letter) returns a hash object, and hash_object.hexdigest() returns the HEX of the
object (i.e, the hash you'd usually see).
```

Now we're going to make a for loop to hash each letter of the alphabet until we find the correct hash using itertools.

Code `#!/usr/bin/env python`

```
import hashlib

def create_SHA256(letter):
    hash_object = hashlib.sha256(letter)
    returnHash = hash_object.hexdigest()
```

```
return returnHash
```

```
hash = raw_input("Hash: ")
alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
```

```
HashFound = False
amountofletters = 1
while not HashFound:
    words = [''.join(i) for i in itertools.product(alphabet, repeat = amountofletters)]
    for a in words:
        if create_SHA256(a) == hash:
            correct = a
            HashFound = True
            break
    amountofletters+=1
```

And then tell the user what the hash was once we found it.

Code `#!/usr/bin/env python`

```
import hashlib
```

```
def create_SHA256(letter):
    hash_object = hashlib.sha256(letter)
    returnHash = hash_object.hexdigest()
    return returnHash
```

```
hash = raw_input("Hash: ")
alphabet = ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z']
```

```
HashFound = False
amountofletters = 1
while not HashFound:
    words = [''.join(i) for i in itertools.product(alphabet, repeat = amountofletters)]
    for a in words:
        if create_SHA256(a) == hash:
            correct = a
            HashFound = True
            break
    amountofletters+=1
```

```
if HashFound:
    print correct
else:
    print "Hash not found"
```

And there we go, we have a SHA-256 bruteforcer.

If you want to test this, I'd suggest only testing 3 letter hashes as 4 letters will take 10 minutes+ and I gave up on 5 letters. This isn't supposed to be an actual cracker, only an introduction to hashlib and itertools.

Now you've learnt something.. You should go congratulate yourself. Go watch some porn :).

nbc_ - December 11, 2014, 7:32 pm

Good introduction, will definitely help people learn. ?

