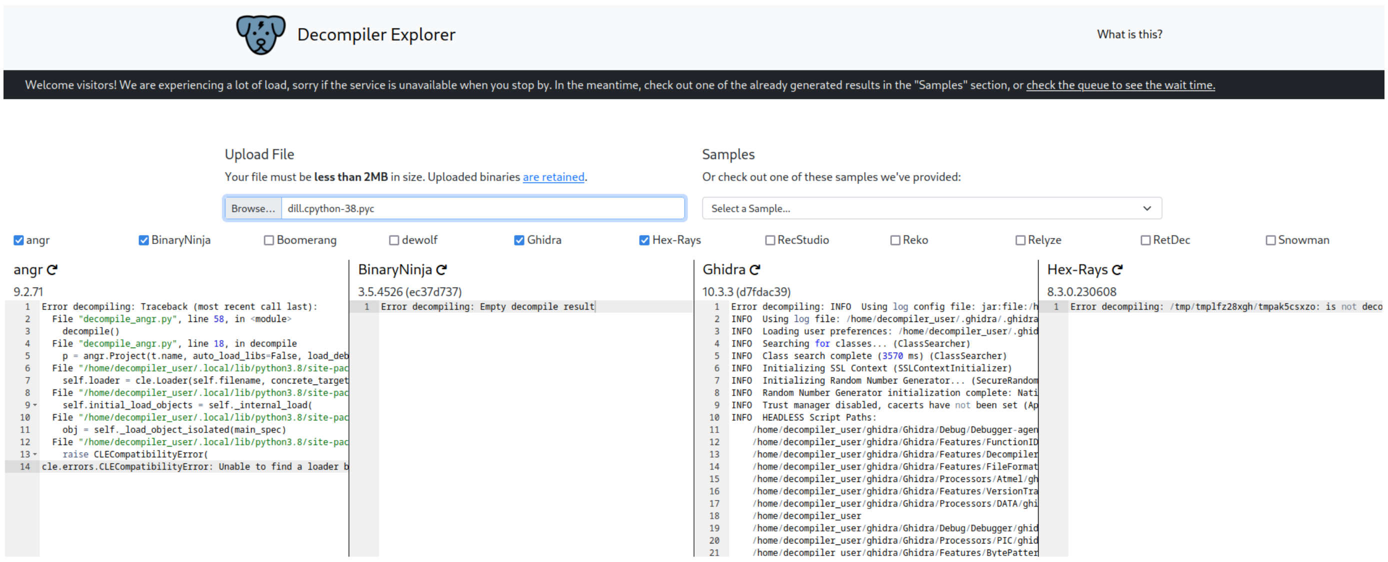
Sunshine-Dill

Dill

> Reversing, "Easy"

"Originally this was going to be about pickles, but .pyc sounds close enough to "pickles" so I decided to make it about that instead."

I spent way too long trying to figure out how to unpickle the file.



Welcome visitors! We are experiencing a lot of load, sorry if the service is unavailable when you stop by. In the meantime, check out one of the already generated results in the "Samples" section, or check the queue to see the wait time.

Upload File
Your file must be **less than 2MB** in size. Uploaded binaries are retained.
Browse... dill.cpython-38.pyc

Samples
Or check out one of these samples we've provided:
Select a Sample...

✓ angr ✓ BinaryNinja ☐ Boomerang ☐ dewolf ✓ Ghidra ✓ Hex-Rays ☐ RecStudio ☐ Reko ☐ Relyze ☐ RetDec ☐ Snowman

angr
9.2.71

```
Error decompiling: Traceback (most recent call last):
  File "decompile_angr.py", line 58, in <module>
    decompile()
  File "decompile_angr.py", line 18, in decompile
    p = angr.Project(t.name, auto_load_libs=False, load_deb
  File "/home/decompiler_user/.local/lib/python3.8/site-pac
    self.loader = cle.Loader(self.filename, concrete_target
  File "/home/decompiler_user/.local/lib/python3.8/site-pac
    self.initial_load_objects = self._internal_load(
  File "/home/decompiler_user/.local/lib/python3.8/site-pac
    obj = self._load_object_isolated(main_spec)
  File "/home/decompiler_user/.local/lib/python3.8/site-pac
    raise CLECompatibilityError(
cle.errors.CLECompatibilityError: Unable to find a loader b
```

BinaryNinja
3.5.4526 (ec37d737)

```
Error decompiling: Empty decompile result
```

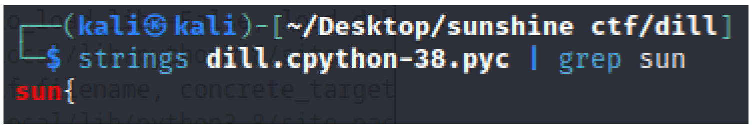
Ghidra
10.3.3 (d7fdac39)

```
Error decompiling: INFO  Using log config file: jar:file:/h
INFO  Using log file: /home/decompiler_user/.ghidra/.ghidra
INFO  Loading user preferences: /home/decompiler_user/.ghid
INFO  Searching for classes... (ClassSearcher)
INFO  Class search complete (3570 ms) (ClassSearcher)
INFO  Initializing SSL Context (SSLContextInitializer)
INFO  Initializing Random Number Generator... (SecureRandom
INFO  Random Number Generator initialization complete: Nati
INFO  Trust manager disabled, cacerts have not been set (Ap
INFO  HEADLESS Script Paths:
    /home/decompiler_user/ghidra/Ghidra/Debug/Debugger-agen
    /home/decompiler_user/ghidra/Ghidra/Features/FunctionID
    /home/decompiler_user/ghidra/Ghidra/Features/Decompiler
    /home/decompiler_user/ghidra/Ghidra/Features/FileFormat
    /home/decompiler_user/ghidra/Ghidra/Processors/Atmel/gh
    /home/decompiler_user/ghidra/Ghidra/Features/VersionTra
    /home/decompiler_user/ghidra/Ghidra/Processors/DATA/ghi
    /home/decompiler_user
    /home/decompiler_user/ghidra/Ghidra/Debug/Debugger/ghid
    /home/decompiler_user/ghidra/Ghidra/Processors/PIC/ghid
    /home/decompiler_user/ghidra/Ghidra/Features/BytePatter
```

Hex-Rays
8.3.0.230608

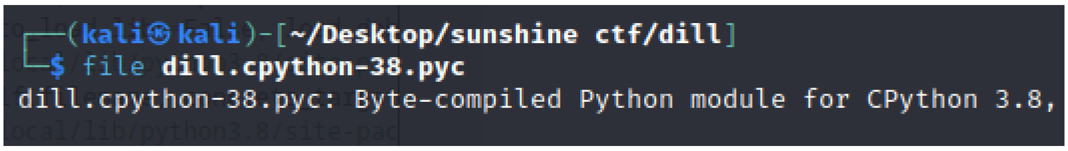
```
Error decompiling: /tmp/tmplfz28xgh/tmpak5csxzo: is not deco
```

Dog bolt showed errors for everything, GPT couldn't figure it out, I was very confused.



```
┌──(kali㉿kali)-[~/Desktop/sunshine ctf/dill]
└─$ strings dill.cpython-38.pyc | grep sun
sun{
```

I knew the flag was partially in the code, because of the strings command.



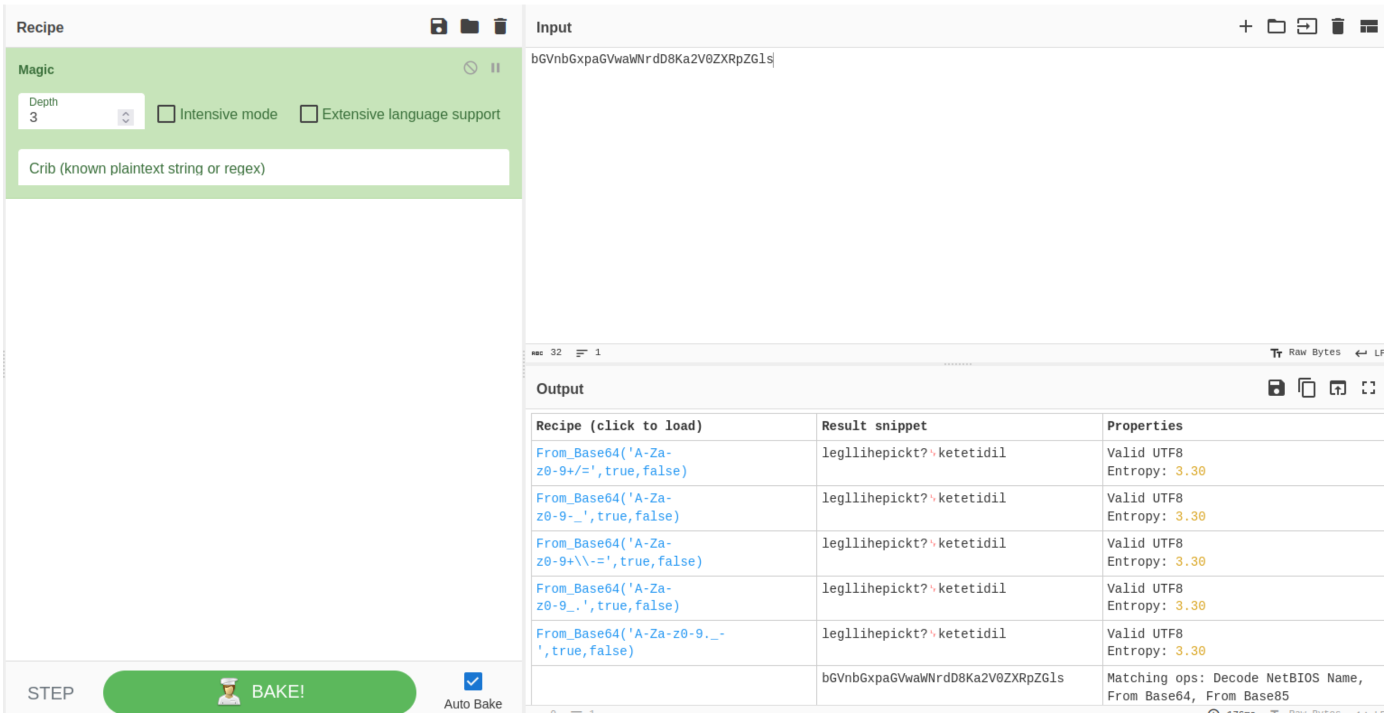
```
┌──(kali㉿kali)-[~/Desktop/sunshine ctf/dill]
└─$ file dill.cpython-38.pyc
dill.cpython-38.pyc: Byte-compiled Python module for CPython 3.8,
```

I ran the file command as well, but didn't realize it at the time. It wasn't a pickle file.

Once I realized this, I searched online '.pyc' decompile online and found the tool: https://www.toolnb.com/tools-lang-en/pyc.html

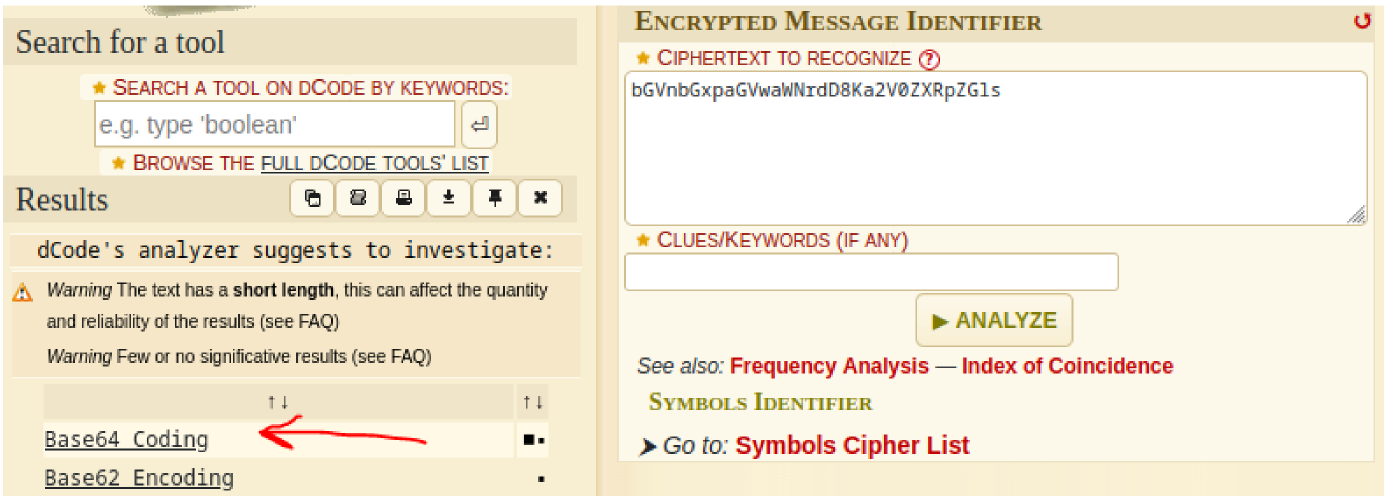
I uploaded the file and got the following code:

```python
# uncompyle6 version 3.5.0
# Python bytecode 3.8 (3413)
# Decompiled from: Python 2.7.5 (default, Jun 20 2023, 11:36:40)
# [GCC 4.8.5 20150623 (Red Hat 4.8.5-44)]
# Embedded file name: dill.py
# Size of source mod 2**32: 914 bytes


class Dill:
    prefix = 'sun{'
    suffix = '}'
    o = [5, 1, 3, 4, 7, 2, 6, 0]

    def __init__(self) -> None:
        self.encrypted = 'bGVnbGxpaGVwaWNrdD8Ka2V0ZXRpZGls'

    def validate(self, value: str) -> bool:
        if not (value.startswith(Dill.prefix) and
value.endswith(Dill.suffix)):
            return False
        value = value[len(Dill.prefix):-len(Dill.suffix)]
        if len(value) != 32:
            return False
        c = [value[i:i + 4] for i in range(0, len(value), 4)]
        value = ''.join([c[i] for i in Dill.o])
        if value != self.encrypted:
            return False
        else:
            return True
```


Recipe
Magic
Depth
3
Intensive mode
Extensive language support
Crib (known plaintext string or regex)
STEP
BAKE!
Auto Bake
Input
Output

```
bGVnbGxpaGVwaWNrdD8Ka2V0ZXRpZGls
```

Recipe (click to load)	Result snippet	Properties
From_Base64('A-Za-z0-9+/=',true,false)	legllihepickt?␊ketetidil	Valid UTF8 Entropy: 3.30
From_Base64('A-Za-z0-9-_',true,false)	legllihepickt?␊ketetidil	Valid UTF8 Entropy: 3.30
From_Base64('A-Za-z0-9+\\-=',true,false)	legllihepickt?␊ketetidil	Valid UTF8 Entropy: 3.30
From_Base64('A-Za-z0-9_.',true,false)	legllihepickt?␊ketetidil	Valid UTF8 Entropy: 3.30
From_Base64('A-Za-z0-9._-',true,false)	legllihepickt?␊ketetidil	Valid UTF8 Entropy: 3.30
	bGVnbGxpaGVwaWNrdD8Ka2V0ZXRpZGls	Matching ops: Decode NetBIOS Name, From Base64, From Base85


Search for a tool
SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'boolean'
BROWSE THE FULL DCODE TOOLS' LIST
Results
ENCRYPTED MESSAGE IDENTIFIER
CIPHERTEXT TO RECOGNIZE
CLUES/KEYWORDS (IF ANY)
ANALYZE
See also: Frequency Analysis — Index of Coincidence
SYMBOLS IDENTIFIER
Go to: Symbols Cipher List

dCode's analyzer suggests to investigate:

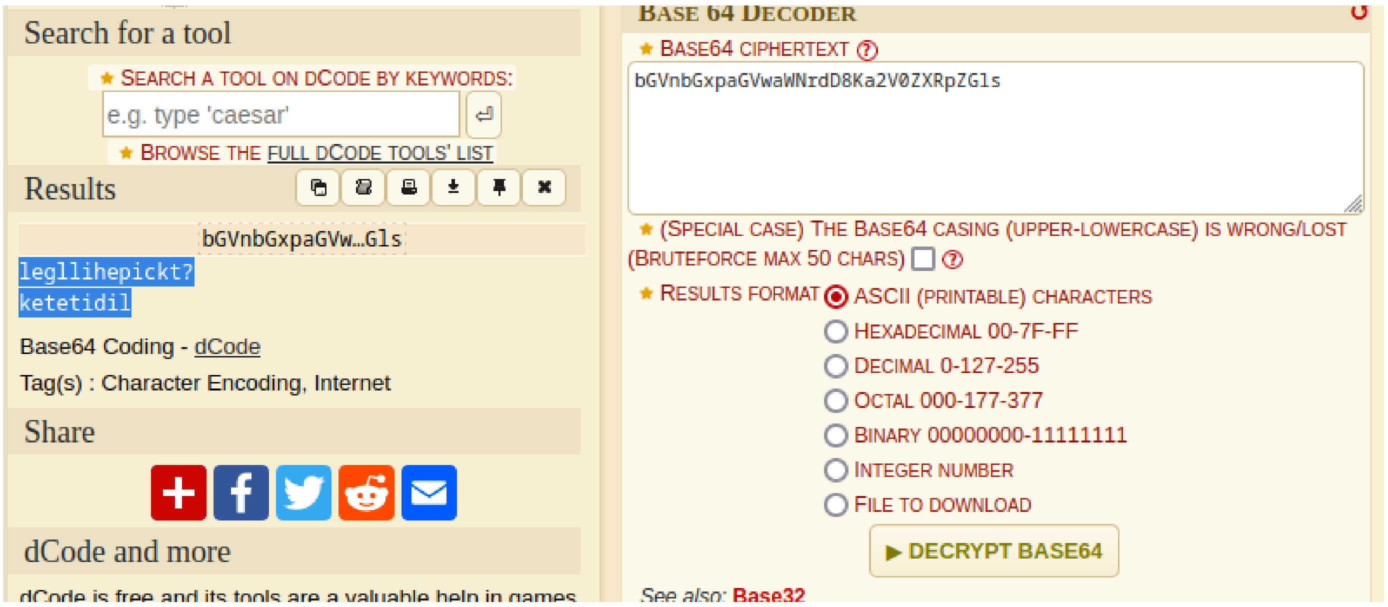
Warning The text has a **short length**, this can affect the quantity and reliability of the results (see FAQ)

Warning Few or no significative results (see FAQ)

Base64 Coding

Base62 Encoding

bGVnbGxpaGVwaWNrdD8Ka2V0ZXRpZGls


Search for a tool
SEARCH A TOOL ON DCODE BY KEYWORDS:
e.g. type 'caesar'
BROWSE THE FULL DCODE TOOLS' LIST
Results
Share
dCode and more
BASE 64 DECODER
BASE64 CIPHERTEXT
(SPECIAL CASE) THE BASE64 CASING (UPPER-LOWERCASE) IS WRONG/LOST (BRUTEFORCE MAX 50 CHARS)
RESULTS FORMAT
DECRYPT BASE64
See also: Base32

bGVnbGxpaGVw...Gls

legllihepickt?
ketetidil

Base64 Coding - dCode

Tag(s) : Character Encoding, Internet

dCode is free and its tools are a valuable help in games

bGVnbGxpaGVwaWNrdD8Ka2V0ZXRpZGls

RESULTS FORMAT [✓ ASCII (PRINTABLE) CHARACTERS] HEXADECIMAL 00-7F-FF DECIMAL 0-127-255 OCTAL 000-177-377 BINARY 00000000-11111111 INTEGER NUMBER FILE TO DOWNLOAD

I spent way too long trying to decrypt the encrypted text, instead of just reading the code.

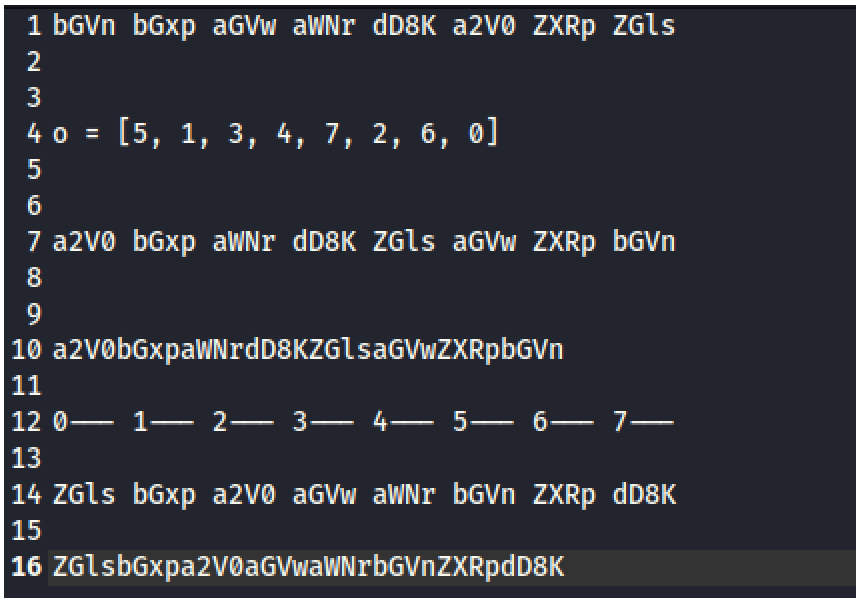
I noticed the validate function was checking to see if the flag matched the encrypted text, so we didn't need to decrypt it anyway.

I noticed the o[] array was used to check segments of 4 chars in the string.

- It was placing the ith segment in the given value into a string to compare to the encrypted value.

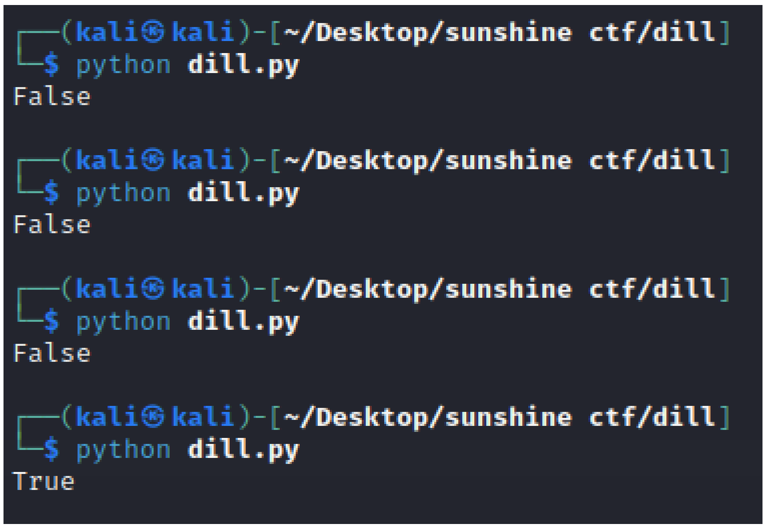
So I segmented the string into its components and mapped the original segments to the positions listed in the o[] array.

- So that when passed through the code, it would be reconstructed to match the encrypted value.



```
bGVn bGxp aGVw aWNr dD8K a2V0 ZXRp ZGls


o = [5, 1, 3, 4, 7, 2, 6, 0]


a2V0 bGxp aWNr dD8K ZGls aGVw ZXRp bGVn


a2V0bGxpaWNrdD8KZGlsaGVwZXRpbGVn

0——  1——  2——  3——  4——  5——  6——  7——

ZGls bGxp a2V0 aGVw aWNr bGVn ZXRp dD8K

ZGlsbGxpa2V0aGVwaWNrbGVnZXRpdD8K
```

This could be automated with python for larger strings.



```
┌──(kali㉿kali)-[~/Desktop/sunshine ctf/dill]
└─$ python dill.py
False

┌──(kali㉿kali)-[~/Desktop/sunshine ctf/dill]
└─$ python dill.py
False

┌──(kali㉿kali)-[~/Desktop/sunshine ctf/dill]
└─$ python dill.py
False

┌──(kali㉿kali)-[~/Desktop/sunshine ctf/dill]
└─$ python dill.py
True
```

I also created a python file that calls this validate function to test if the flag was correct.

```python
class Dill:
    prefix = 'sun{'
    suffix = '}'
    o = [5, 1, 3, 4, 7, 2, 6, 0]

    def __init__(self) -> None:
        self.encrypted = 'bGVnbGxpaGVwaWNrdD8Ka2V0ZXRpZGls'

    def validate(self, value: str) -> bool:
        if not (value.startswith(Dill.prefix) and
value.endswith(Dill.suffix)):
            return False
        value = value[len(Dill.prefix):-len(Dill.suffix)]
        if len(value) != 32:
            return False
        c = [value[i:i + 4] for i in range(0, len(value), 4)]
        value = ''.join([c[i] for i in Dill.o])
        if value != self.encrypted:
            return False
        else:
            return True

dill = Dill()

print( dill.validate('sun{ZGlsbGxpa2V0aGVwaWNrbGVnZXRpdD8K}') )
```

After getting a True back from the function, I tested the flag on the Sunshine CTF and it worked!