WRITEUP

CSCG 2024

CHALLANGE: CAN I HAZ LOLPYTHON?

From: werter (Discord name)

Date: 31.03.2024

CONTENT

1. THE CHALLENGE

Name: Can I haz LOLPython?

Authors: 0X4D5A

Categories: MISC

Difficulty: Easy

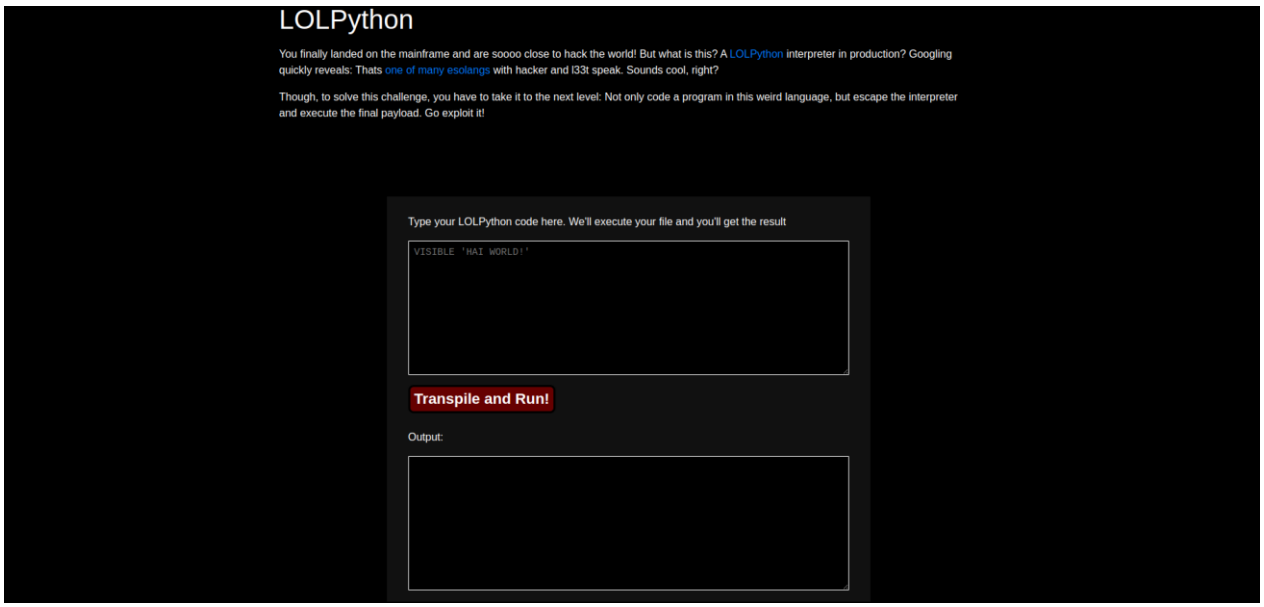
Description: You landed on the mainframe, but now face the final boss: LOLPython!

Given: A website and a zip directory

Summarize: The challenge is to write a script in a joke programming language named LOLPython. This script needs to read the flag from the server and give it back.

2. THE WEBSITE

To solve the challenge access to a website is given.



On the top of the Website is a short help text.

> You finally landed on the mainframe and are soooo close to hack the world! But what is this? A LOLPython interpreter in production? Googling quickly reveals: Thats one of many esolangs with hacker and l33t speak. Sounds cool, right?
>
> Though, to solve this challenge, you have to take it to the next level: Not only code a program in this weird language but escape the interpreter and execute the final payload. Go exploit it!

Under this text is a box where LOLPython code can be written. Then a button with the text “Transpile and Run!” follows. If this button is pressed the code will be executed and the output stands in the bottom box.

3. THE CODE

In the given zip directory are multiple files and directories. The tree view:

```
.
├── apache-config.conf
├── Dockerfile
├── flag
└── src
    ├── css
    │   ├── bootstrap.css
    │   ├── bootstrap-grid.css
    │   ├── bootstrap-grid.min.css
    │   ├── bootstrap.min.css
    │   ├── bootstrap-reboot.css
    │   ├── bootstrap-reboot.min.css
    │   └── custom.css
    ├── form.php
    ├── index.php
    ├── js
    │   ├── bootstrap.bundle.js
    │   ├── bootstrap.bundle.min.js
```

```
│   ├── bootstrap.js
│   ├── bootstrap.min.js
│   └── rockstar.js
└── transpile.php
```

apache-config.conf

contains the configuration for the Apache webserver. The information's are not important.

Dockerfile

The Dockerfile starts the webserver. It contains installations and configurations for the challenge. It can be used to create a local test server. This file gives us information's over the location of the flag on the server.

flag

Contains the flag on the server. Here is only a test flag.

src/css/bootstrap*

CSS files from the bootstrap framework can be ignored.

src/css/custome.css

Contains custom CSS can be ignored.

src/js/bootstrap*

JavaScrip files from the bootstrap framework can be ignored.

src/js/rockstar.js

Contains a function transpil. Get executed if the button on the website gets pressed. Send the contend from the top box to the transpil.php on server as Json. The response gets printed in the bottom box.

src/form.php

Contains the html code for the help text and the form with the two boxes and the button.

src/index.php

Contains the contend of the side we see. Include the other scripts.

src/transpil.php

Is the central file of the challenge. It saves the content sent from the rockstar.js file in a /tmp/lolpython_prog_ file. This file gives the LOLPython compiler to be executed. The output is sent back.

4. WHAT IS LOLPYTHON

Because LOLPython play in this challenge a important rule it is useful to know something over it.

LOLPython is a joke programming language. It is not made to be really used. It is developed from Andrew Dalker. The Compiler translate the code to python. This python code gets normally executed. The language is inspirated from the more famous LOLCode language.

5. SOLUTION

As the help text say we must write a script in this language. To write a script in a new language it can be useful to read the Doku from the language. The problem is:

I don’t find a Doku, but I found the code from the compiler in GitHub.

https://github.com/KartikTalwar/LOLPython

After an analysis i found out that every python keyword, operator and more, used by python, gets replaced by some other word.

if with IZ

self with ME

brake with KTHXBYE

[...]

This means I can write a script in python to read the flag and need only to translate this to LOLPython with the information's from the compiler.

How has the python script to look like?

There are multiple ways to solve this problem. Like read the file. But I prefer to write a shell to execute commands on the server. This gives me more freedom.

So could it look like In python:

```
import os
os.system('id')
```

Now the only thing left is the translation.

Replace:

import with GIMME

. with OWN

(with WIT

) with OK

The finale code looks:

```
GIMME os
os OWN system WIT 'id' OK
```

When I run this I get the response:

uid=33(www-data) gid=33(www-data) groups=33(www-data)

To get the flag the only thing to replace is the command.

```
GIMME os
os OWN system WIT 'cat /flag' OK
```

The Output is the FLAG

6. FIX VULNERABILITY!

There are multiple configurations that can be done to prevent damage. In the following part I give some examples.

1. **Filtering / Blacklisting**

It is possible to filter specific commands. That can be dangerous. If one command is detected it won't be executed. A full filtering would be hard to implement. Functions of the language that are supposed to be presented or taught would not longer work. This is the reason why you can still create shells or read files on servers from learning page like w3schools. There are better ways to have the functionality and still be save.

2. **Sandboxing**

The CSCG uses docker for the challenges for sandboxing and prevents any damage on their own servers. Other forms can be used like VMs. In a sandbox should be only important functions. No important data. A break out is possible but very hard. It is also possible to spawn a private sandbox for every user which would be destroyed after the user leave.

3. **Rights**

You can run scripts with different rights. The use of Root Rights is clearly not secure. In Linux it is also possible to set different CAPs. Specifically, rights like CAP_CHOWN or CAP_NET_ADMIN and other should be deactivated.

4. **Limited Packages**

In python and other languages is possible to install and remove packages. This packages can have dangerous code. Like os I used above. The problem is that some packages are system relevant and have important code like os. On learning pages like w3schools they often play an important rule.

5. **Resource and time limits**

Resource and time limits can be used to prevent DOS attacks. Users can create big resource and time intensive task. Or accidentally create an infinity loop. This can destroy the service.

6. **Logging**

The logging of the access is useful to found out what happening when something happened. IP addresses can be used to identify attackers.

7. **Compile on user side.**

In some languages is possible to execute the script on the user side. This prevents any damages to the server. This is possible in languages like html CSS or JS.

8. **Firewall**

A firewall can be useful to block access from specific IP addresses. It can be useful to prevent ping backs and other forms of connections instead of https.

9. **User authentication**

A user system can prevent not authenticated connections to the server. A not allowed use can be very fast traced back to a user.