AISecurity-Research-Template

Guangtao Zhang

2023-12-27

Table of contents

1. Overview

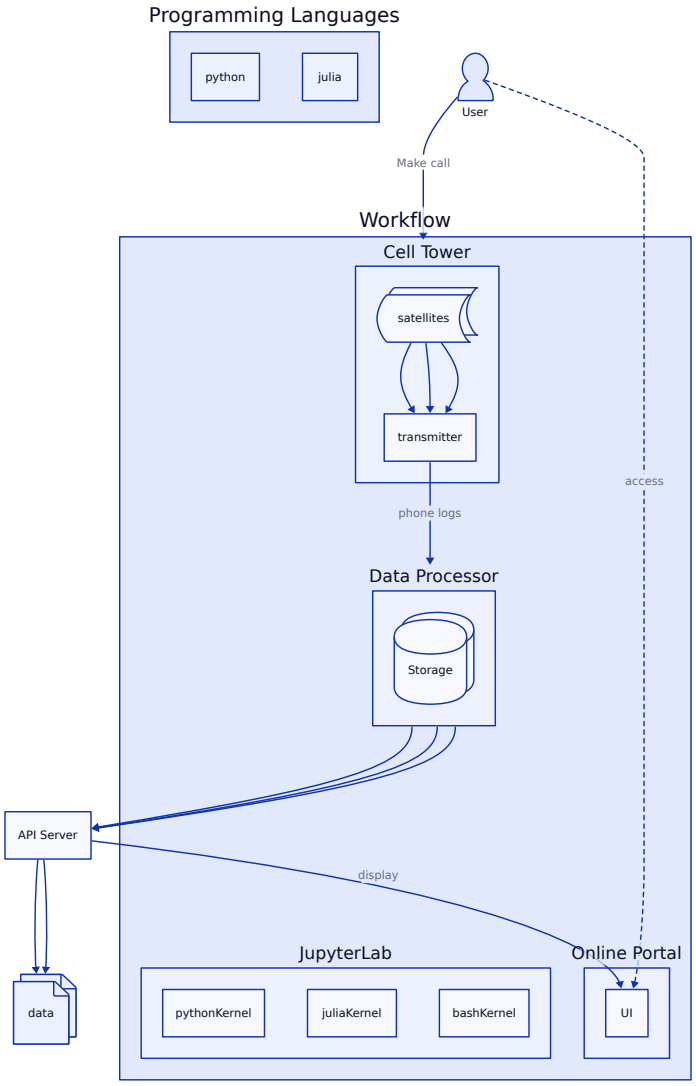
2. Getting Stared

```
git clone
 ↪  https://github.com/hardenedlinux/AISecurity-Research-Template
cd AISecurity-Research-Template

docker pull
 ↪  ghcr.io/hardenedlinux/aisecurity-research-template
docker run --platform linux/amd64 -it -v "$(pwd):/work"
 ↪  --entrypoint bash
 ↪  ghcr.io/hardenedlinux/aisecurity-research-template:latest

podman run --platform linux/amd64 -it -v "$(pwd):/work"
 ↪  --userns=keep-id --entrypoint bash
 ↪  ghcr.io/hardenedlinux/aisecurity-research-template-users:latest
```

- Actions:
  - jupyterlab: just jupyterlab-run
  - quarto: just quarto example

3. Workflow

```
from IPython.core.display import SVG
import os

os.environ["PATH"] += os.pathsep + "$PATH"
SVG(filename='flow.svg')
```


Programming Languages
python
julia
User
Make call
Workflow
Cell Tower
satellites
transmitter
access
phone logs
Data Processor
Storage
API Server
display
JupyterLab
Online Portal
data
pythonKernel
juliaKernel
bashKernel
UI

4. Acknowledgments

- Guidelines for secure AI system development

Part I.

Julia

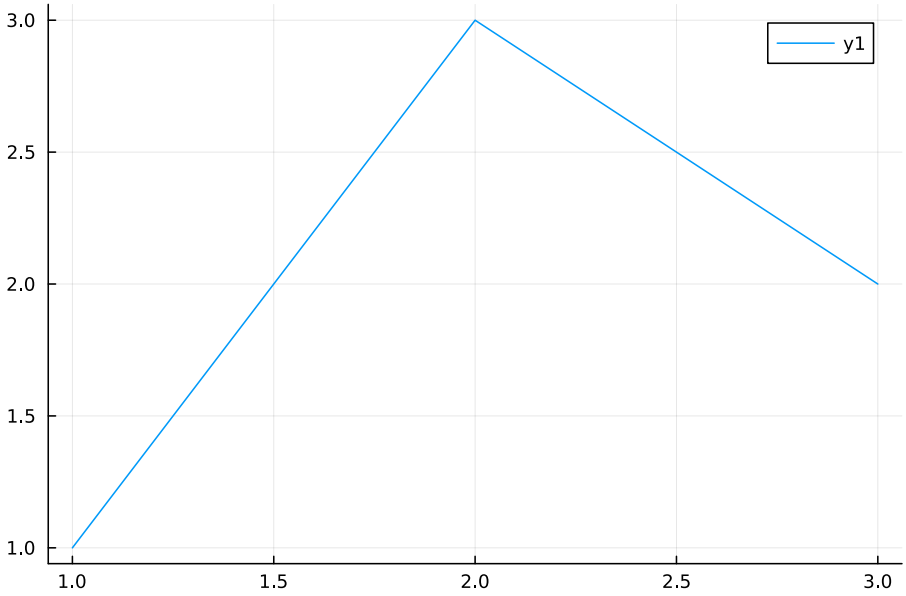
- https://julialang.org/
- https://cn.julialang.org/

5. Julia Plots Test

5.1. Plots

```
using Plots
```

```
display(plot([1,3,2]))
```



5.2. Makie

- https://docs.makie.org/stable/tutorials/basic-tutorial/

```
using GLMakie # All functionality is defined in Makie
 ↪  and every backend re-exports Makie

Base.@kwdef mutable struct Lorenz
    dt::Float64 = 0.01
     ::Float64 = 10
     ::Float64 = 28
     ::Float64 = 8/3
    x::Float64 = 1
    y::Float64 = 1
    z::Float64 = 1
end

function step!(l::Lorenz)
    dx = l.  * (l.y - l.x)
    dy = l.x * (l.  - l.z) - l.y
    dz = l.x * l.y - l.  * l.z
    l.x += l.dt * dx
    l.y += l.dt * dy
    l.z += l.dt * dz
    Point3f(l.x, l.y, l.z)
end

attractor = Lorenz()

points = Observable(Point3f[]) # Signal that can be used
 ↪  to update plots efficiently
colors = Observable(Int[])
```

```
set_theme!(theme_black())

fig, ax, l = lines(points, color = colors,
    colormap = :inferno, transparency = true,
    axis = (; type = Axis3, protrusions = (0, 0, 0, 0),
              viewmode = :fit, limits = (-30, 30, -30,
 ↪  30, 0, 50)))

record(fig, "lorenz.mp4", 1:120) do frame
    for i in 1:50
        # update arrays inplace
        push!(points[], step!(attractor))
        push!(colors[], frame)
    end
    ax.azimuth[] = 1.7pi + 0.3 * sin(2pi * frame / 120)
 ↪  # set the view angle of the axis
    notify(points); notify(colors) # tell points and
     ↪  colors that their value has been updated
    l.colorrange = (0, frame) # update plot attribute
 ↪  directly
end
```

```
"lorenz.mp4"
```

```
using Base64

function display_mp4(filename)
    display("text/html", string("""<video autoplay
     ↪  controls><source
     ↪  src="data:video/x-m4v;base64,""",
```

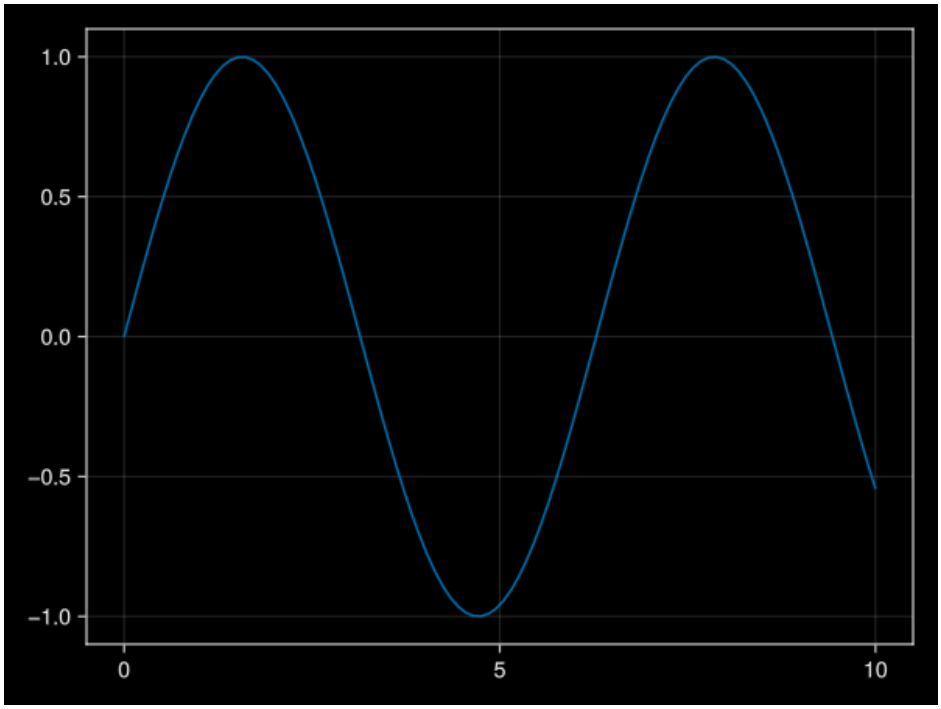
```
    Base64.base64encode(open(read,filename)),"""
     ↪  type="video/mp4"></video>"""))
end
```

```
display_mp4 (generic function with 1 method)
```

```
display_mp4("lorenz.mp4")
```

```
Unable to display output for mime type(s): text/html
```

```
f = Figure()
ax = Axis(f[1, 1])
x = range(0, 10, length=100)
y = sin.(x)
lines!(ax, x, y)
f
```


1.0
0.5
0.0
−0.5
−1.0
0
5
10

Part II.

Python

6. OpenAIPy Test

6.1. openAPI Python binding

```
from scipy.special import comb
```

```
com = comb(5, 2, exact = False, repetition=True)
```

```
com
```

```
15.0
```

```
import os
from openai import OpenAI

client = OpenAI(api_key = os.getenv("OPENAI_API_KEY"))
```

```
completion = client.chat.completions.create(
  model="gpt-4",
  messages=[
    {"role": "system", "content": "You are a Security
↪   Engineer.)"},
```

```
  ]
)
```

```
import textwrap
def wrap_text(text, width):
    lines = text.split('\n')
    wrapped_lines = []
    for line in lines:
        if len(line) > width:
            wrapped_lines.extend(textwrap.wrap(line,
 ↪  width=width))
        else:
            wrapped_lines.append(line)
    return '\n'.join(wrapped_lines)

print(wrap_text(completion.choices[0].message.content,70))
```

```
As a Security Engineer, my primary responsibilities include:

1. Developing and implementing security protocols and procedures
across all IT departments.
2. Regularly auditing the company's systems and network, identifying
possible threats or vulnerabilities.
3. Ensuring data and network security are maintained at all times.
4. Installing, configuring, and supporting security tools such as
firewalls, anti-virus software, and patch management systems.
5. Keeping abreast of the latest developments in IT security and
ensuring that the organization responds swiftly to new security
threats.
6. Conducting both routine and irregular security checks for any
possible network breach.
```

7. Carrying out risk assessments and make recommendations for improvement.
8. Ensuring compliance with relevant security-related regulations and protocols.
9. Responding promptly and effectively to any security incidents.
10. Providing training and guidance to colleagues on information security matters.
11. Documenting any security breaches and assessment the damage they might have caused.
12. Implementing strategies to lessen the risk of future security breaches.