Security Attack and Defense Reasoning Framework (SADRF)

STUDENTS: YU-CHUN FANG, KAMERON VUONG, SARA SHIN, RUOQI ZHANG

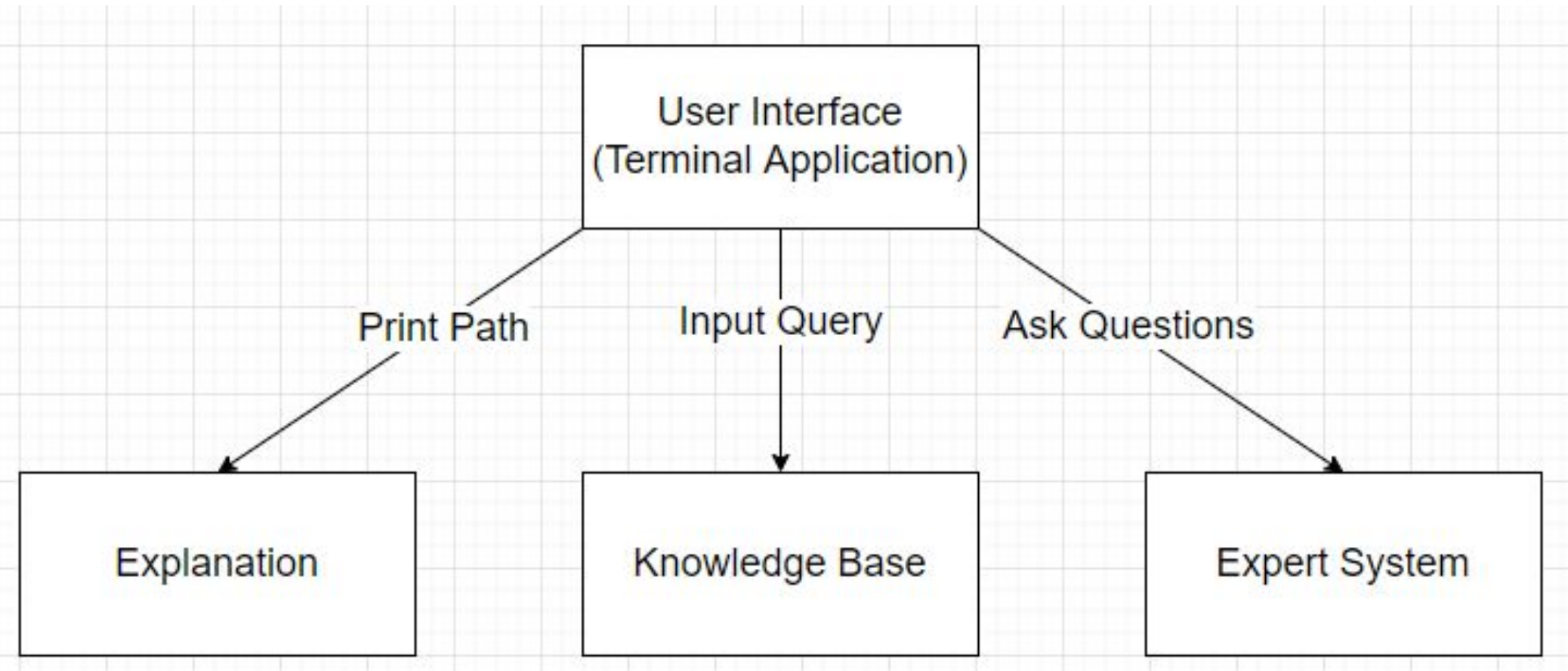
07

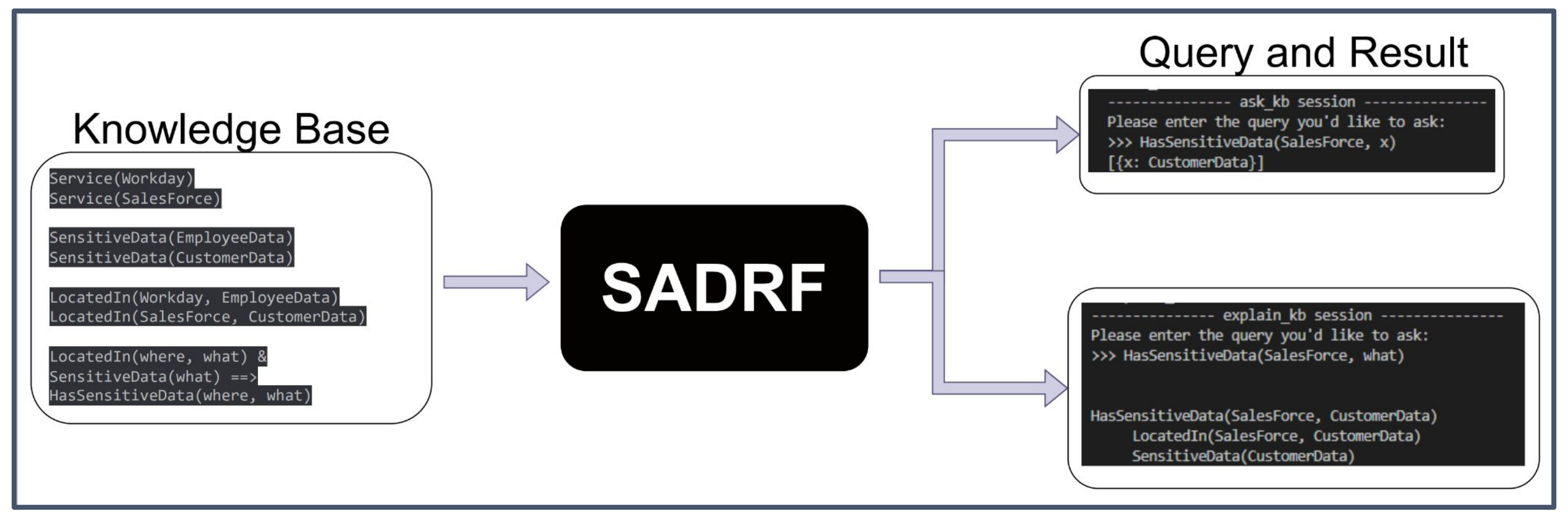
The Problem

- Cybersecurity is a constantly evolving field. Even though the goals and motivations of attackers remain relatively unchanged, the techniques and methodologies used by attackers change as technologies shift.
- Current security systems look for specific vulnerabilities and thereby are strict and inflexible.
- As a result, the number of common vulnerabilities and exposures (CVEs) has increased every year.

Our Solution

- Develop an expert system that intelligently reasons and infers the potential weakness for file storage
- Uses First Order Logic to make inferences
- Users can create their own unique knowledge base about their system and query the system to find weak areas





```
Service(Workday)
Service(SalesForce)

SensitiveData(EmployeeData)
SensitiveData(CustomerData)

LocatedIn(Workday, EmployeeData)
LocatedIn(SalesForce, CustomerData)

LocatedIn(where, what) &
SensitiveData(what) ==>
HasSensitiveData(where, what)
```

```
--------------- ask_kb session ---------------
Please enter the query you'd like to ask:
>>> HasSensitiveData(SalesForce, x)
[{x: CustomerData}]
```

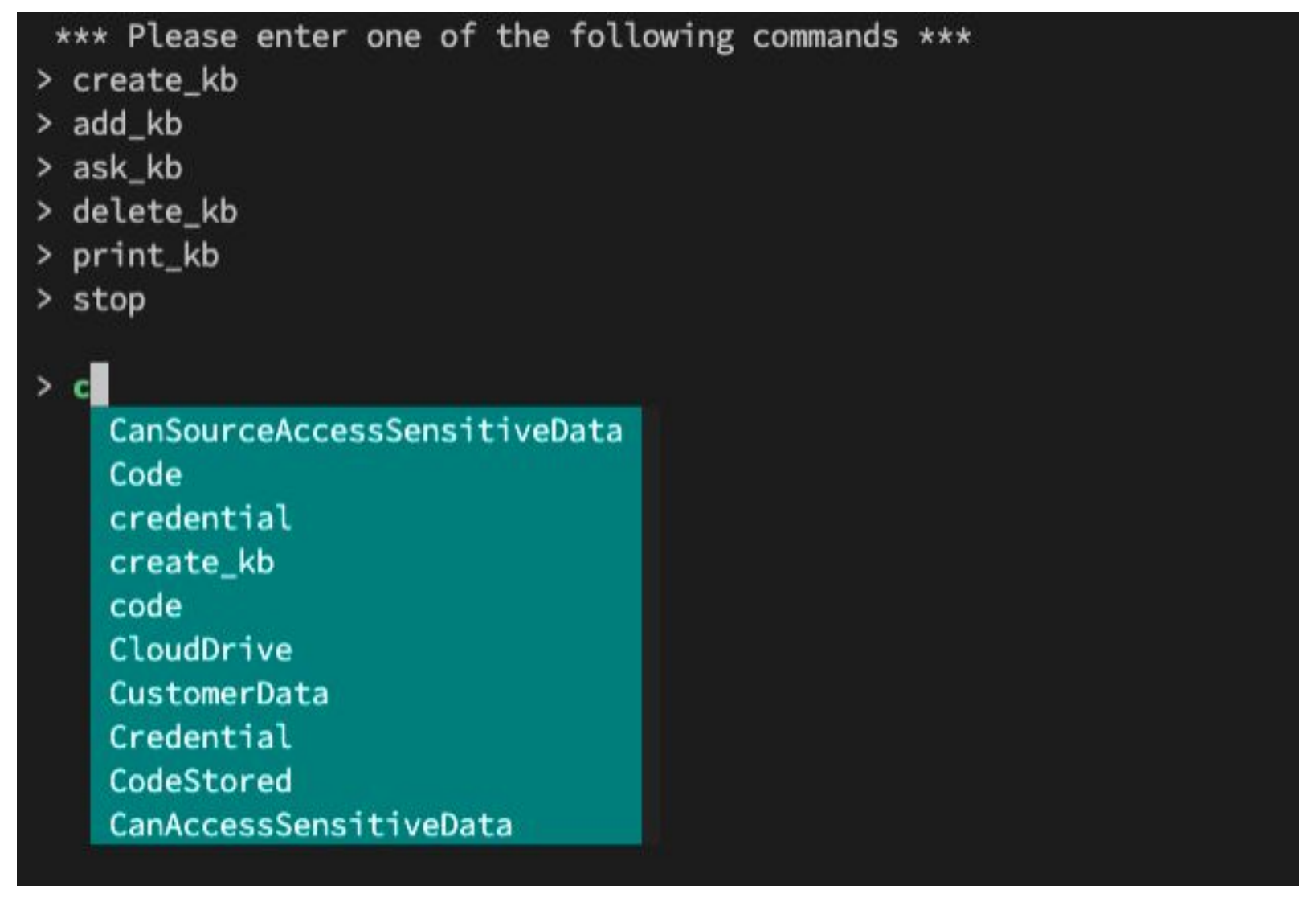
```
--------------- explain_kb session ---------------
Please enter the query you'd like to ask:
>>> HasSensitiveData(SalesForce, what)


HasSensitiveData(SalesForce, CustomerData)
    LocatedIn(SalesForce, CustomerData)
    SensitiveData(CustomerData)
```

AIMA Python: Logic Notebooks

- The open-source python library, aimacode-python implements first order logic in python [1]
- It additionally provides python implementations of first order logic algorithm from Russell and Norvig's "Artificial Intelligence - A Modern Approach" [2]
- From this library, the forward chaining and backward chaining algorithms are used in the implementation of the first order logic expert system

Terminal Application

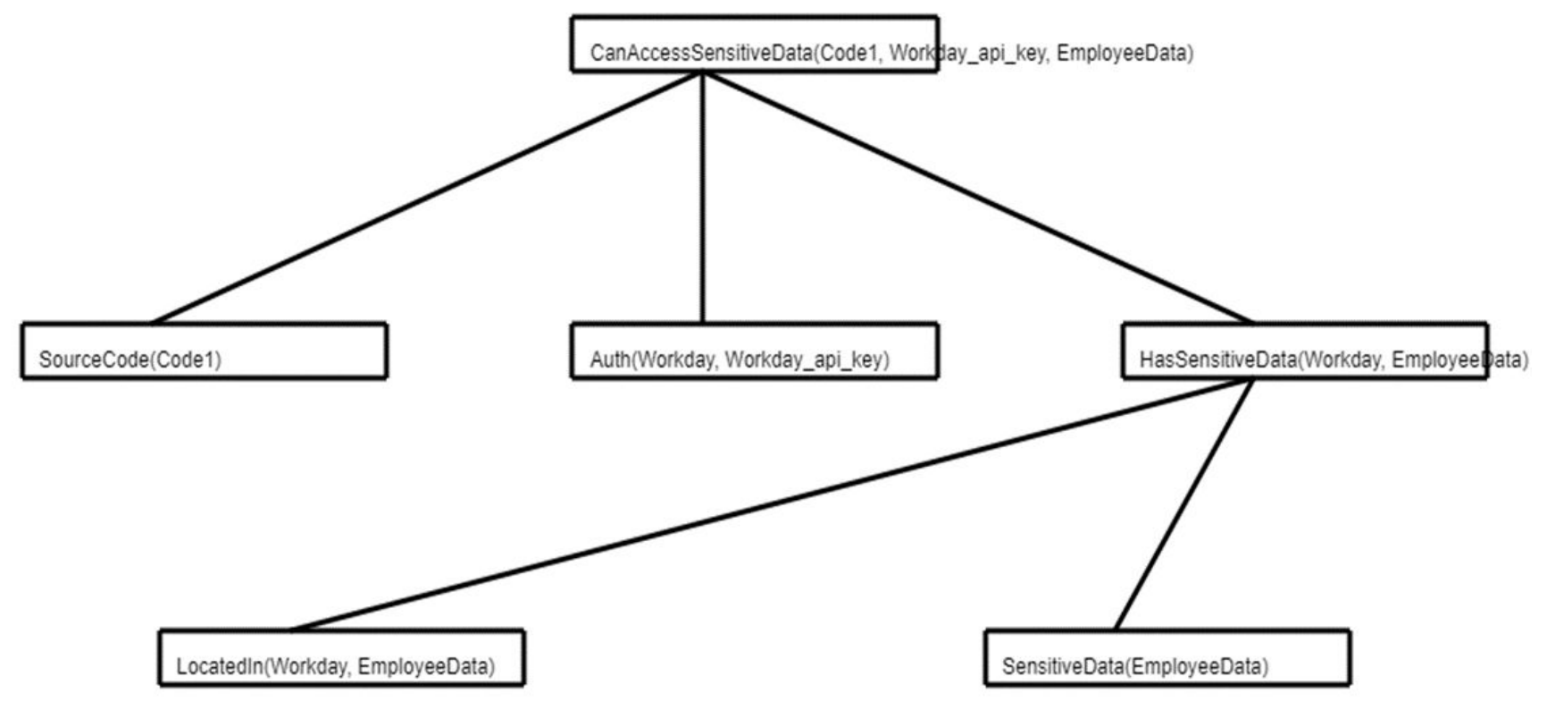
- Users can
  - Create a knowledge base
  - Add queries to the knowledge base
  - Display all queries in the current knowledge base
  - Delete queries from the knowledge base
  - Ask questions of the knowledge base
- Complete with auto-complete function for easy user input

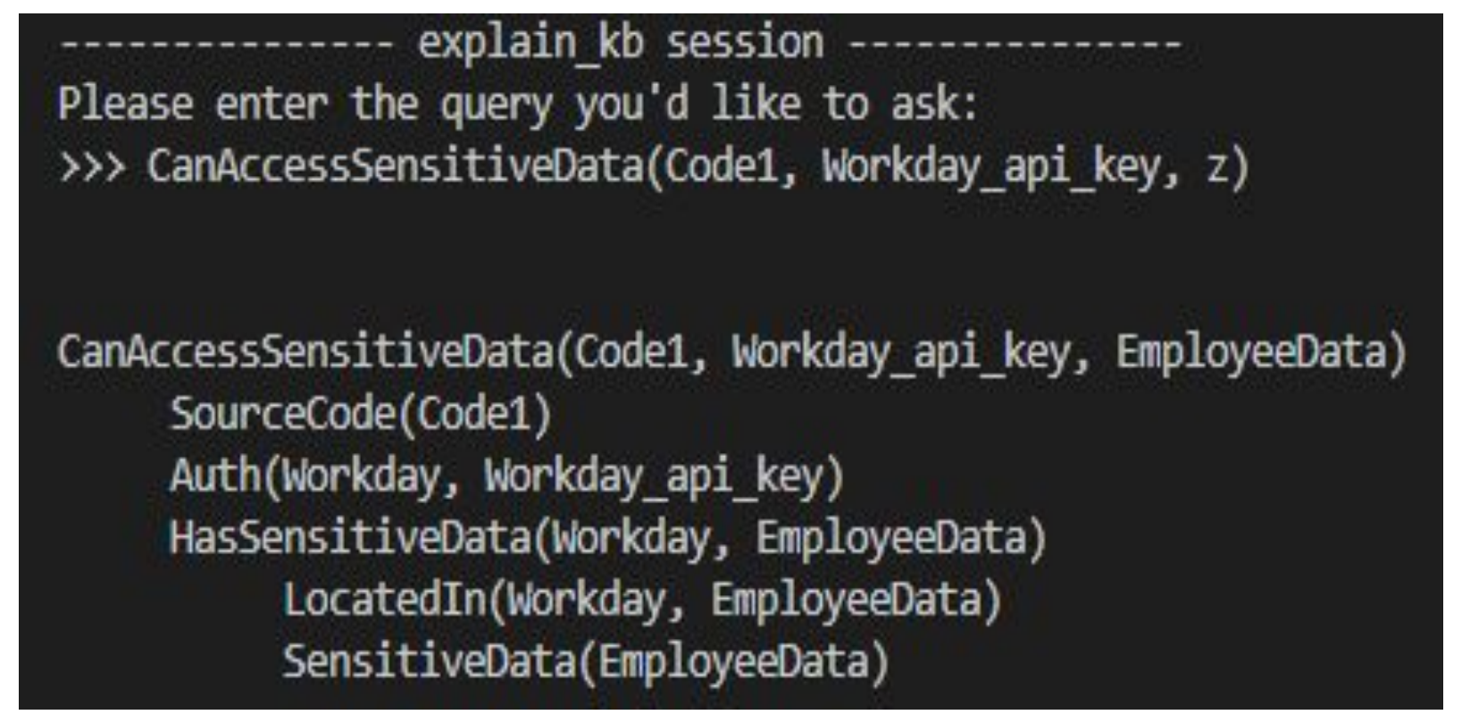


```
 *** Please enter one of the following commands ***
> create_kb
> add_kb
> ask_kb
> delete_kb
> print_kb
> stop

> c
   CanSourceAccessSensitiveData
   Code
   credential
   create_kb
   code
   CloudDrive
   CustomerData
   Credential
   CodeStored
   CanAccessSensitiveData
```

Reasoning/Visualization

- In addition to returning a result to the user's query, the expert system returns a textual reasoning/explanation for why it returns the result.

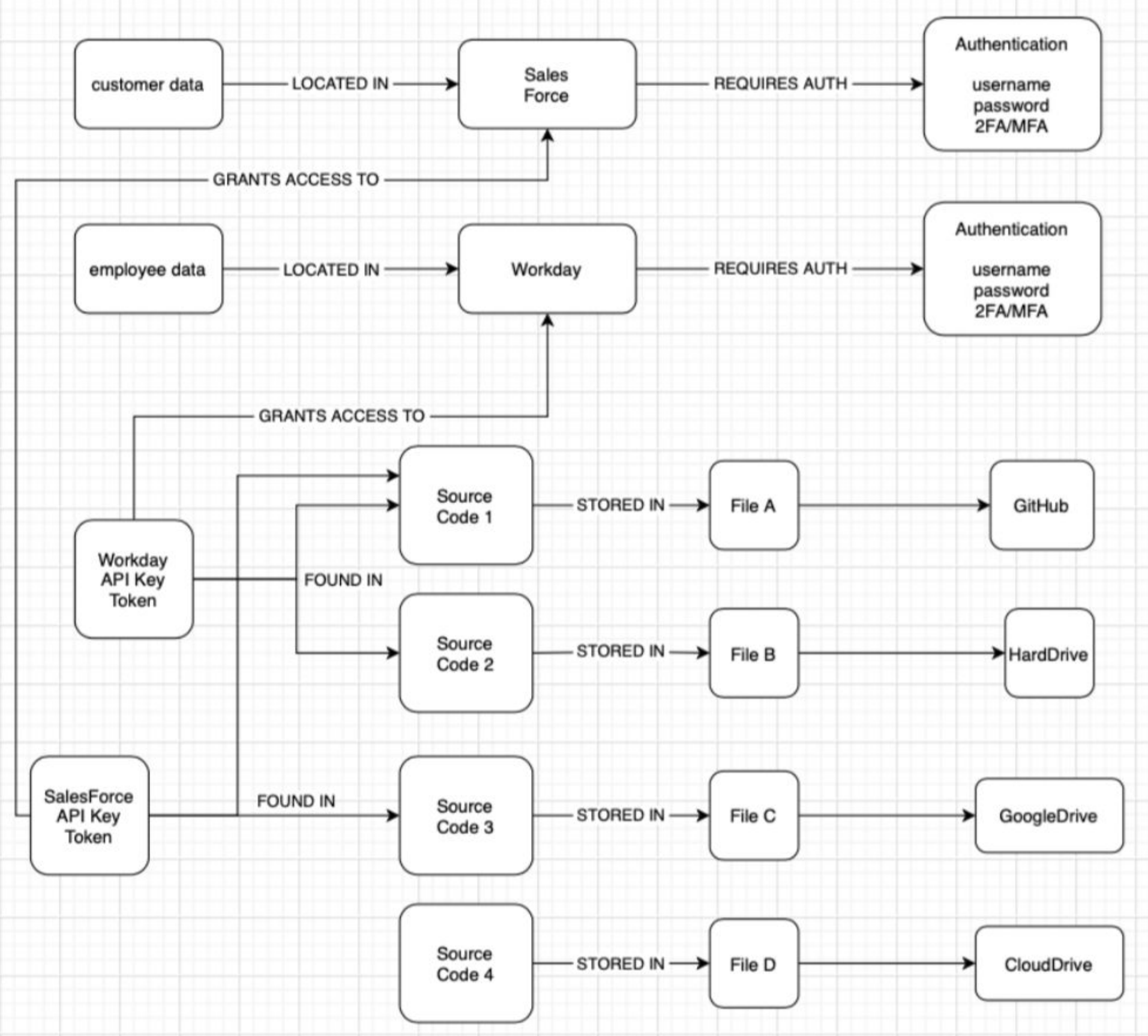




```
--------------- explain_kb session ---------------
Please enter the query you'd like to ask:
>>> CanAccessSensitiveData(Code1, Workday_api_key, z)


CanAccessSensitiveData(Code1, Workday_api_key, EmployeeData)
    SourceCode(Code1)
    Auth(Workday, Workday_api_key)
    HasSensitiveData(Workday, EmployeeData)
        LocatedIn(Workday, EmployeeData)
        SensitiveData(EmployeeData)
```

Knowledge Base Explanation

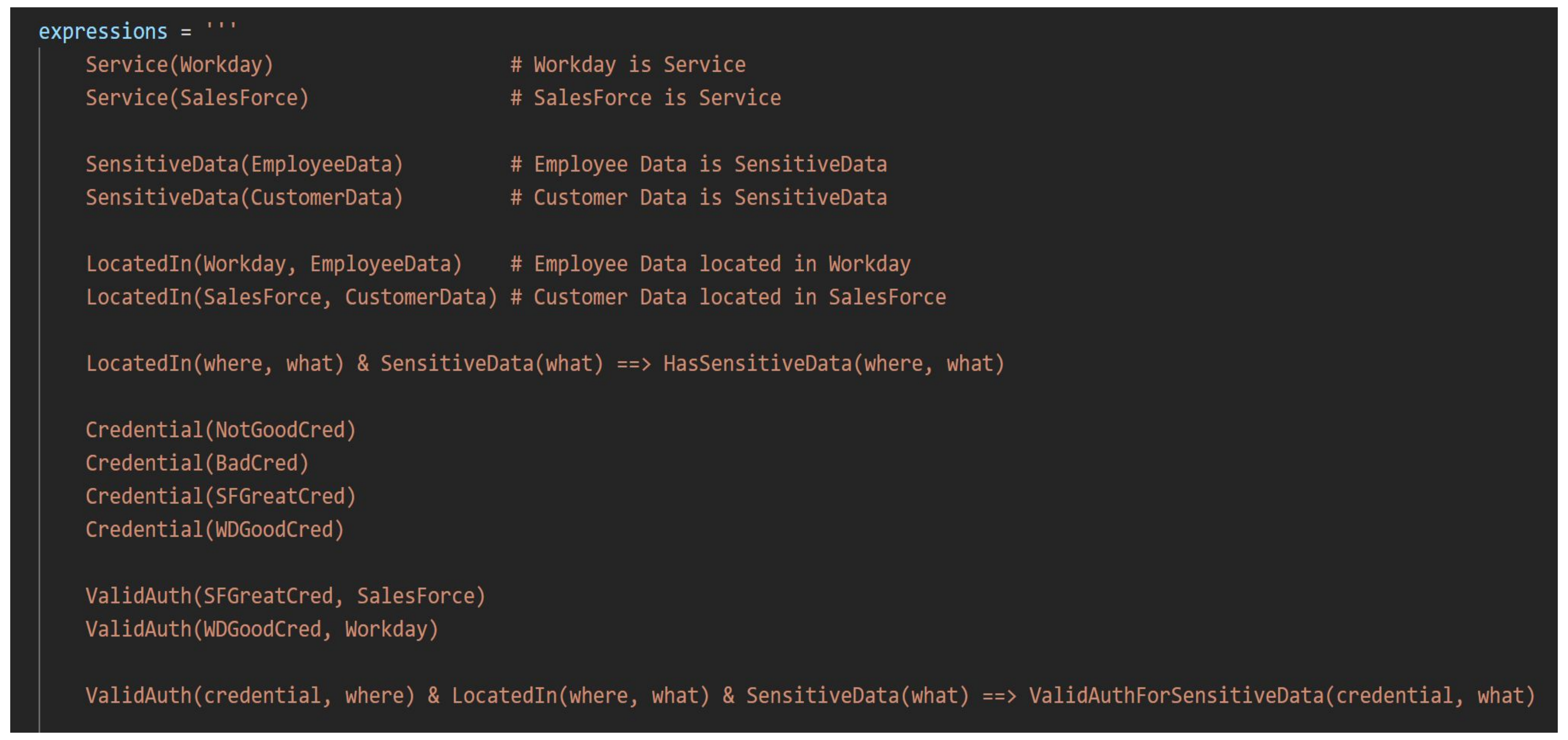
- The knowledge base describes the paths how files/documents stored
- Two sensitive data: Employee Data & Customer Data
- The sensitive data can be accessed via two paths: Auth / File - Drive



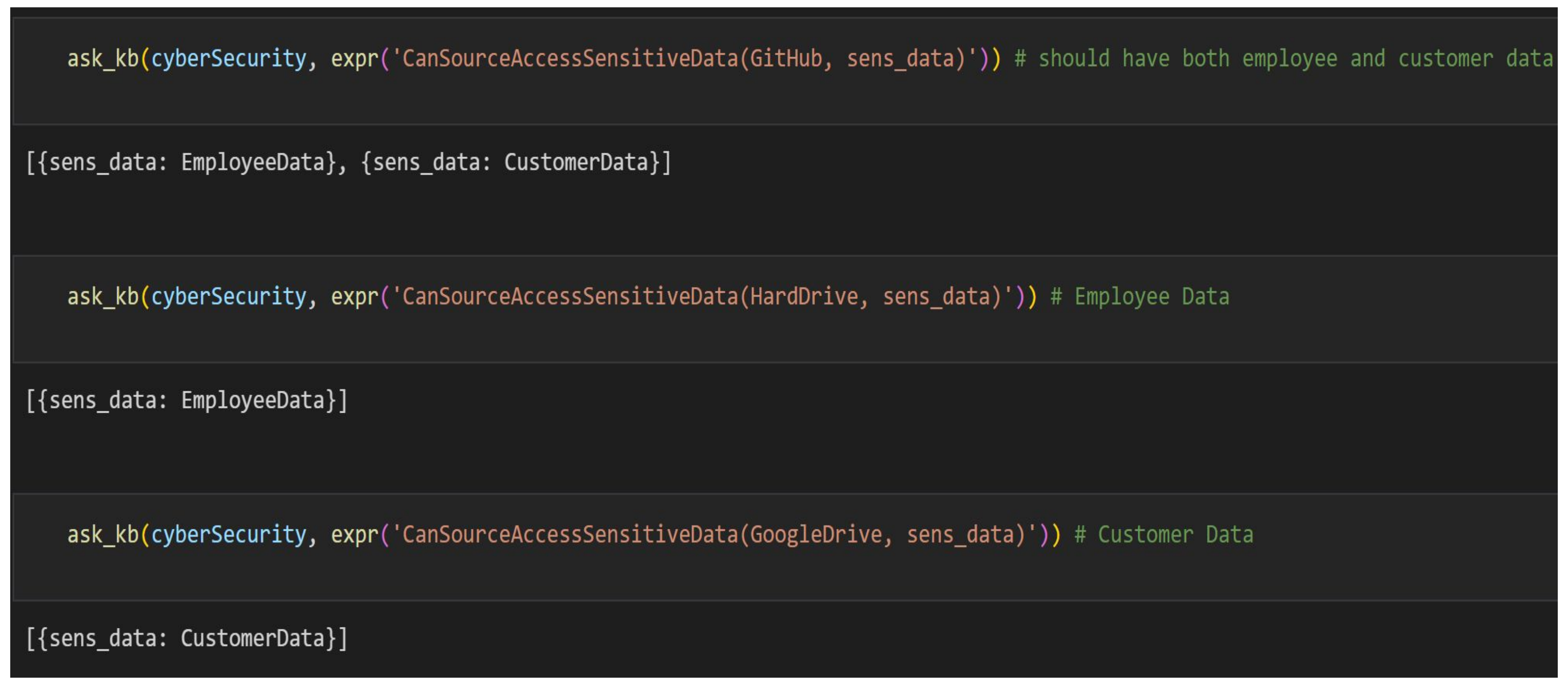
Knowledge Base Implementation

Manually enter the

- Type of object
- Storage location
- Relevant path
- Testing which locations have access to sensitive data



```
expressions = '''
    Service(Workday)                    # Workday is Service
    Service(SalesForce)                 # SalesForce is Service

    SensitiveData(EmployeeData)         # Employee Data is SensitiveData
    SensitiveData(CustomerData)         # Customer Data is SensitiveData

    LocatedIn(Workday, EmployeeData)    # Employee Data located in Workday
    LocatedIn(SalesForce, CustomerData) # Customer Data located in SalesForce

    LocatedIn(where, what) & SensitiveData(what) ==> HasSensitiveData(where, what)

    Credential(NotGoodCred)
    Credential(BadCred)
    Credential(SFGreatCred)
    Credential(WDGoodCred)

    ValidAuth(SFGreatCred, SalesForce)
    ValidAuth(WDGoodCred, Workday)

    ValidAuth(credential, where) & LocatedIn(where, what) & SensitiveData(what) ==> ValidAuthForSensitiveData(credential, what)
```



```
    ask_kb(cyberSecurity, expr('CanSourceAccessSensitiveData(GitHub, sens_data)')) # should have both employee and customer data

[{sens_data: EmployeeData}, {sens_data: CustomerData}]


    ask_kb(cyberSecurity, expr('CanSourceAccessSensitiveData(HardDrive, sens_data)')) # Employee Data

[{sens_data: EmployeeData}]


    ask_kb(cyberSecurity, expr('CanSourceAccessSensitiveData(GoogleDrive, sens_data)')) # Customer Data

[{sens_data: CustomerData}]
```

Future Work, References, and Acknowledgments

- Develop more intuitive UI for easier use
- Turning first order logic queries input into natural language queries
  - SensitiveData(EmployeeData)
  - → Employee Data is Sensitive data
  - CanSourceAccessSensitiveData(GitHub, sens_data)
  - →What sensitive data can GitHub access?
- Reimplementing the visualization result to image connected graph for easier to understand

[1] AIMAPython, https://github.com/aimacode/aima-python

[2] Stuart Russell, Peter Norvig "Artificial Intelligence: A Modern Approach, 3rd Edition", University of California at Berkeley, Pearson, 2010

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY *of* WASHINGTON

INDUSTRY MENTOR: TERRY RODERY

SPONSORS: NUTANIX, Inc.

FACULTY MENTOR: JOHN RAITI

TEACHING ASSISTANT: SHRUTI MISRA