Python - Ldap Injection

Running the app on Docker

```
$ sudo docker pull blabla1337/owasp-skf-lab:ldap-injection
```

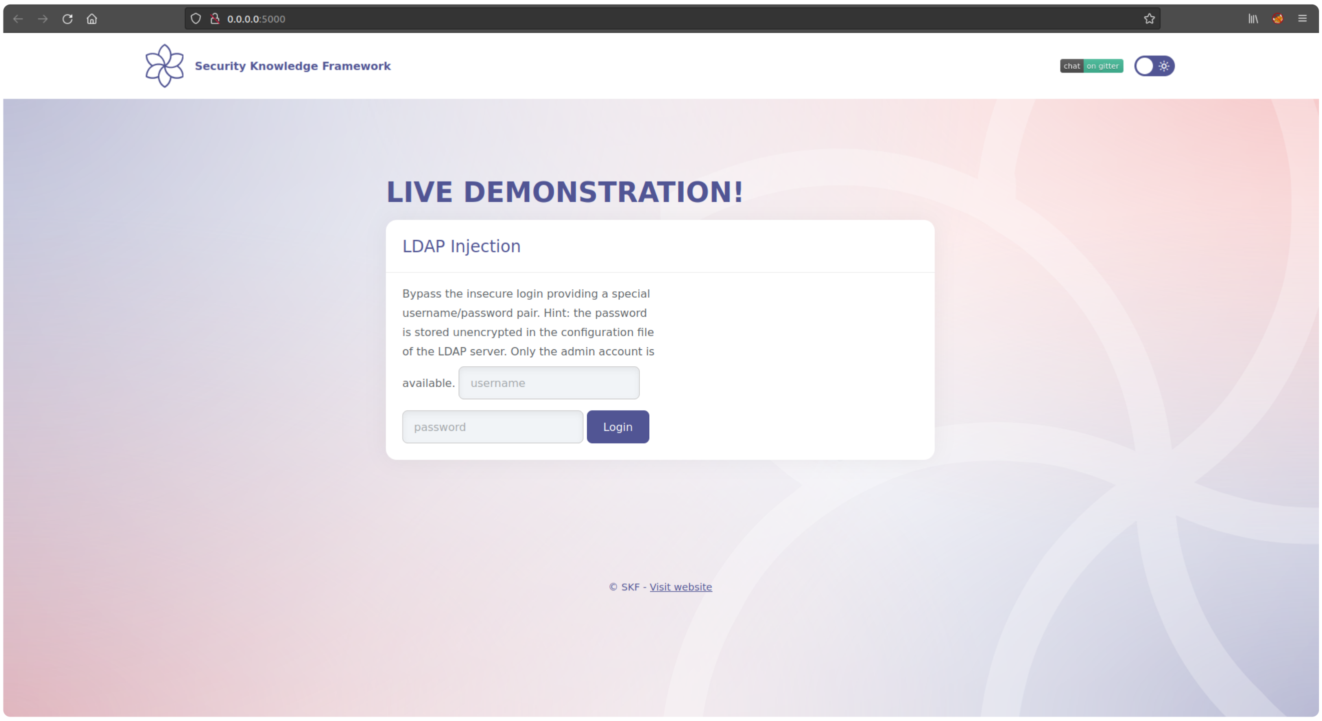
```
$ sudo docker run -ti -p 127.0.0.1:5000:5000 blabla1337/owasp-skf-lab:ldap-injection
```

> ✓ Now that the app is running let's go hacking!

Reconnaissance

LDAP Injection is an attack used to exploit web based applications that construct LDAP statements based on user input. When an application fails to properly sanitize user input, it’s possible to modify LDAP statements using a local proxy. This could result in the execution of arbitrary commands such as granting permissions to unauthorized queries, and content modification inside the LDAP tree. The same advanced exploitation techniques available in SQL Injection can be similarly applied in LDAP Injection.

Let's open the app.

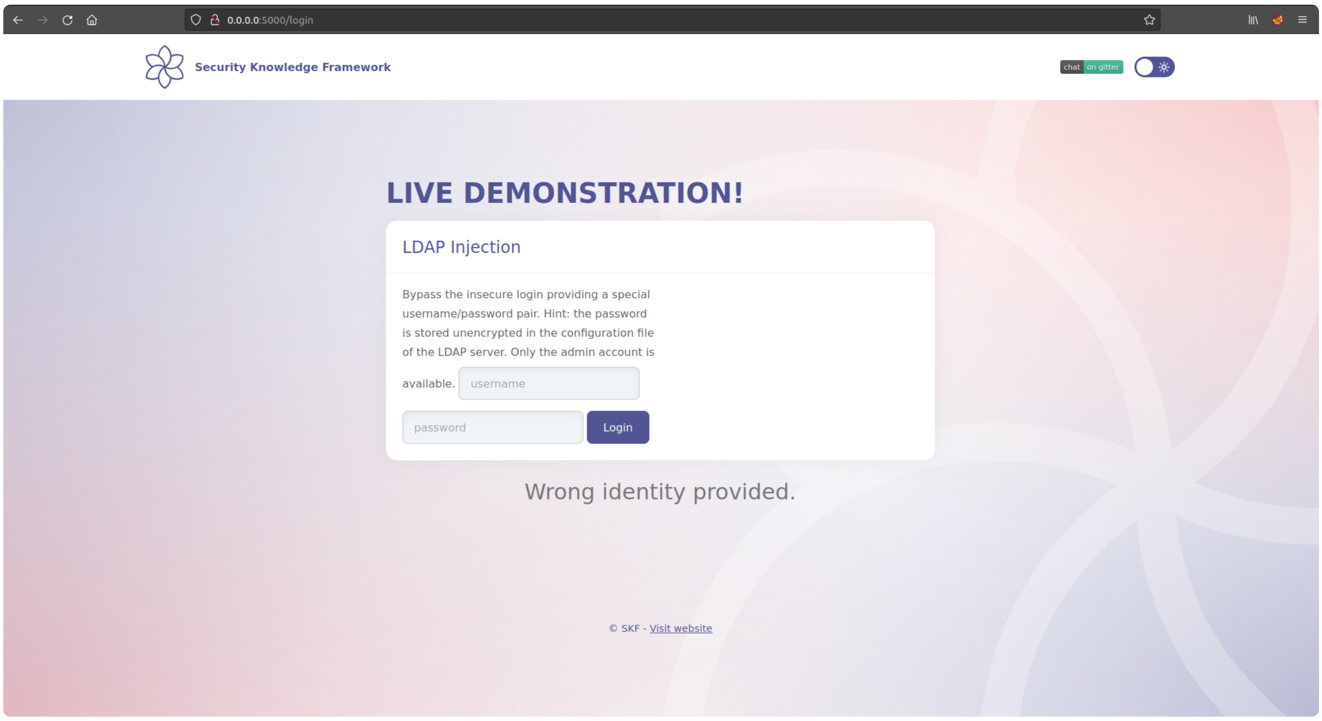


LIVE DEMONSTRATION!

LDAP Injection

Bypass the insecure login providing a special username/password pair. Hint: the password is stored unencrypted in the configuration file of the LDAP server. Only the admin account is available.

Trying to loggin with a random username and password.



LIVE DEMONSTRATION!

LDAP Injection

Bypass the insecure login providing a special username/password pair. Hint: the password is stored unencrypted in the configuration file of the LDAP server. Only the admin account is available.

Wrong identity provided.

The application architecture that supports LDAP includes both server-side and client-side components. The LDAP queries submitted to the server are known as LDAP search filters, which are constructed using prefix notation. Below is an example of an LDAP search filter:

```
find("(&(cn=" + username +")(userPassword=" + pass +"))")
```

This prefix filter notation instructs the query to find an LDAP node with the given username and password.

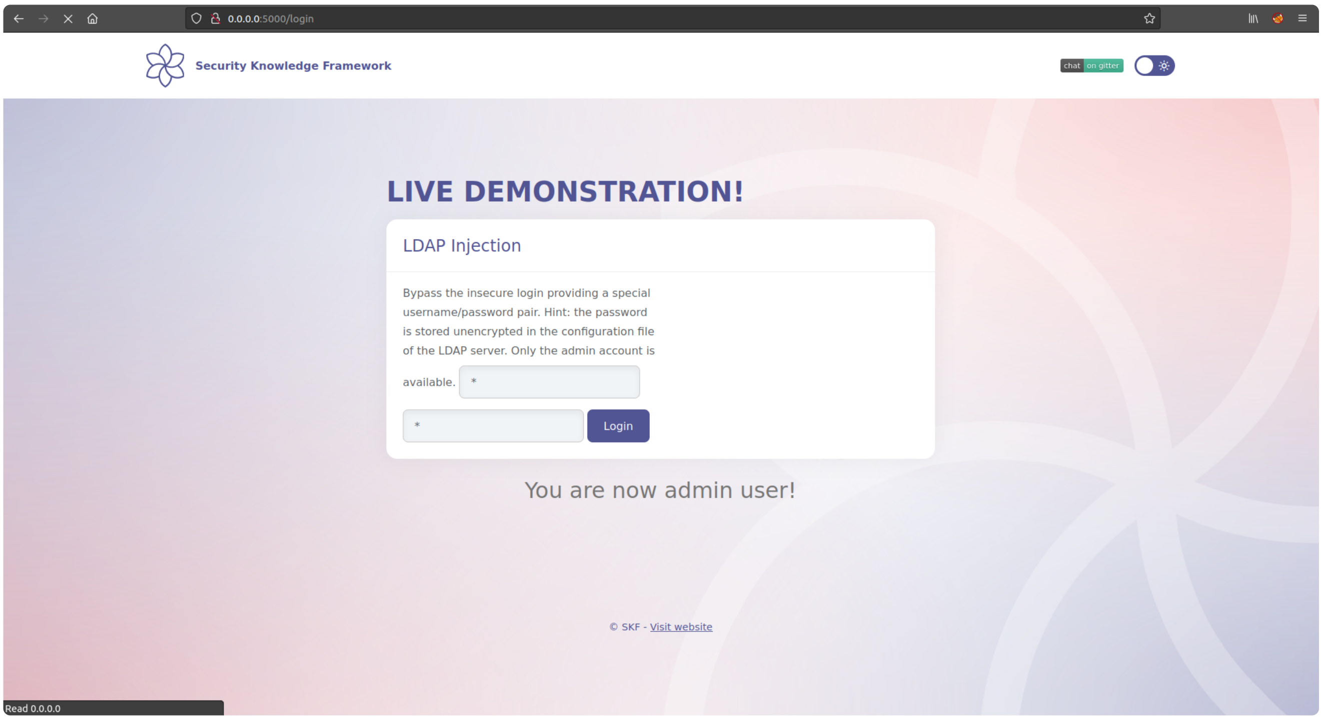
Exploitation

Let's check ldap-injection.py.

```
search_filter = "(&(cn="+username+")(sn="+password+"))"
```

We can see that the filter is constructed by concatenating the username and password directly into the filter without proper sanitization. If we replace the username and password with a special character we can bypass authentication controls. Using * as the username and password will result in a successful login.

```
search_filter = "(&(cn=" * ")(sn=" * "))";
```



LIVE DEMONSTRATION!

LDAP Injection

Bypass the insecure login providing a special username/password pair. Hint: the password is stored unencrypted in the configuration file of the LDAP server. Only the admin account is available. *

*

Login

You are now admin user!

We successfully logged in as the Admin.

Additional sources

LDAP Injection | OWASP Foundation

LDAP Injection Prevention - OWASP Cheat Sheet Series

What Is LDAP Injection and How Does It Work? | Synopsys
synopsys