SECURE CODE WARRIOR

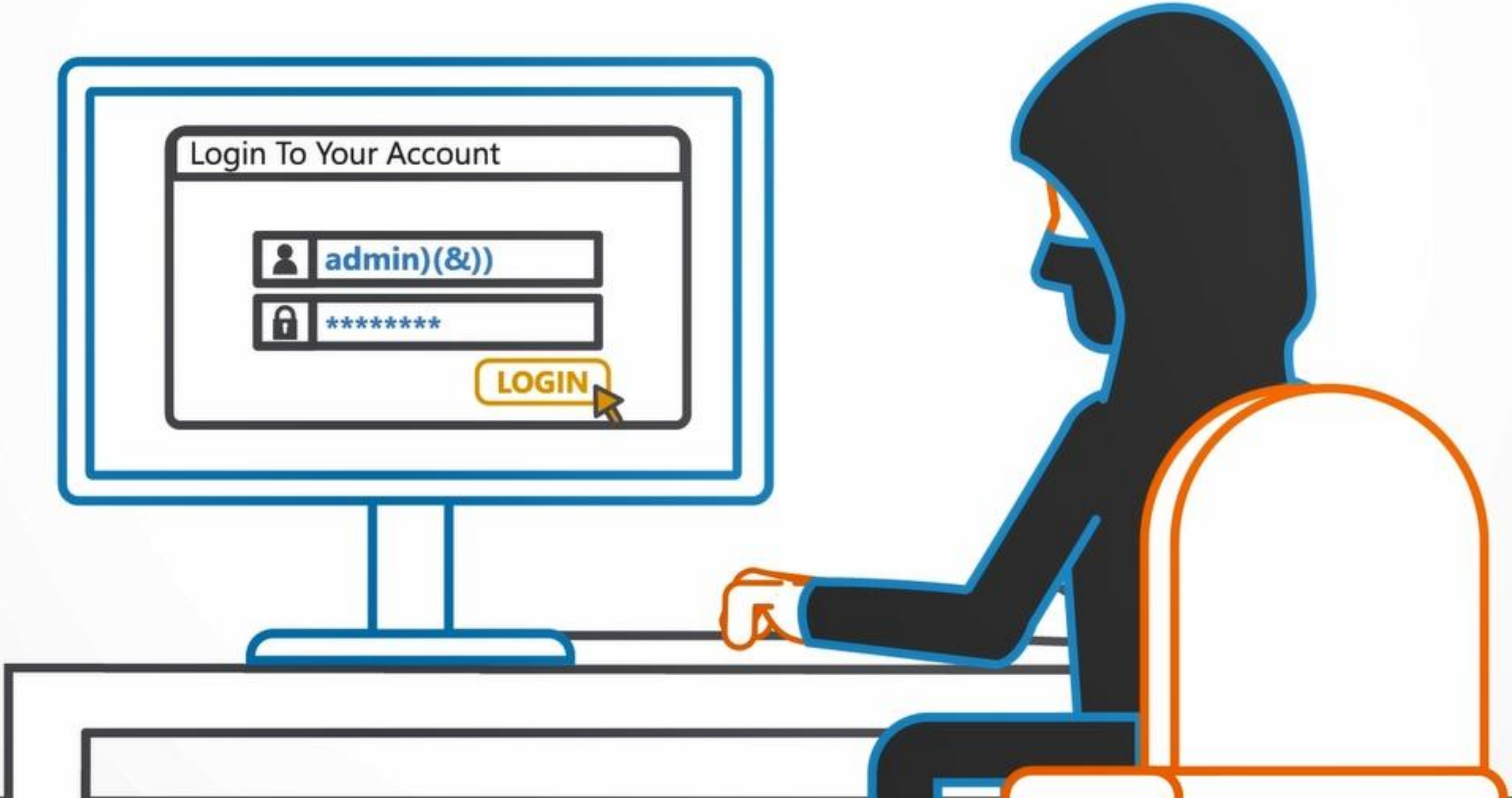
LDAP INJECTION

We will explain

what LDAP Injection is,
its causes and preventions and some potential hazards

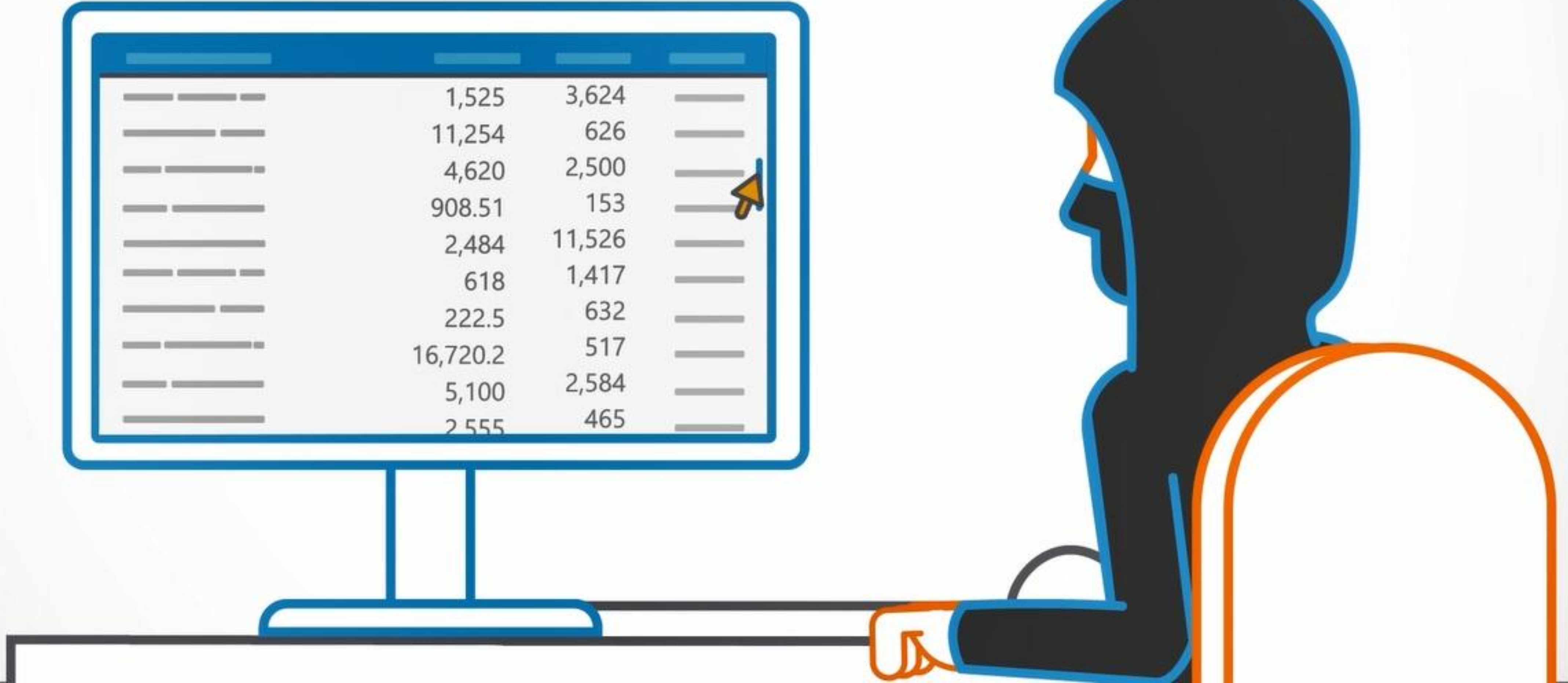
An “LDAP Injection” is a vulnerability by which an attacker can influence back-end

SO, WHAT IS AN LDAP INJECTION ?



LDAP queries by injecting malicious LDAP statements, via user controllable input.

SO, WHAT IS AN LDAP INJECTION ?

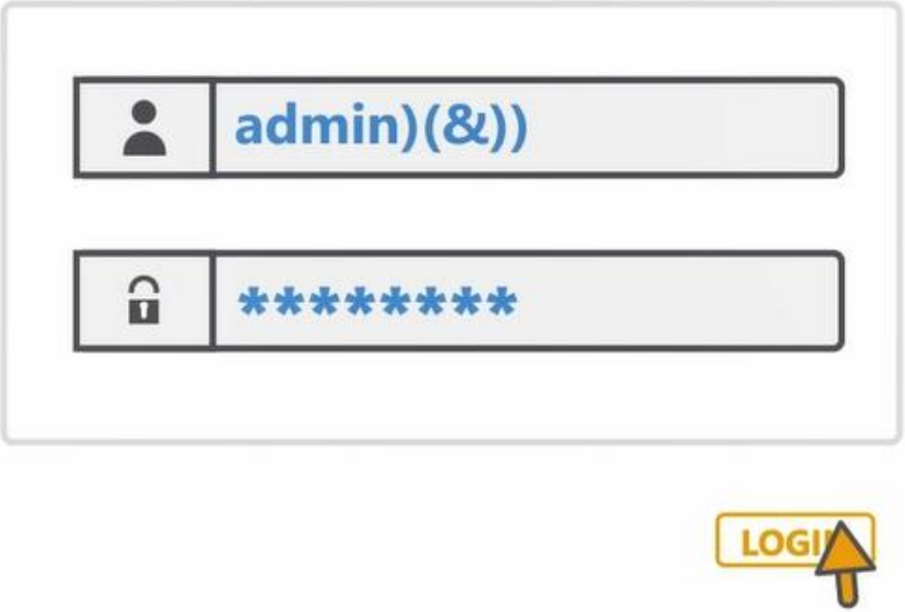


1,525	3,624
11,254	626
4,620	2,500
908.51	153
2,484	11,526
618	1,417
222.5	632
16,720.2	517
5,100	2,584
2,555	465

WHAT CAUSES AN LDAP INJECTION?

User input is used to dynamically build LDAP queries.

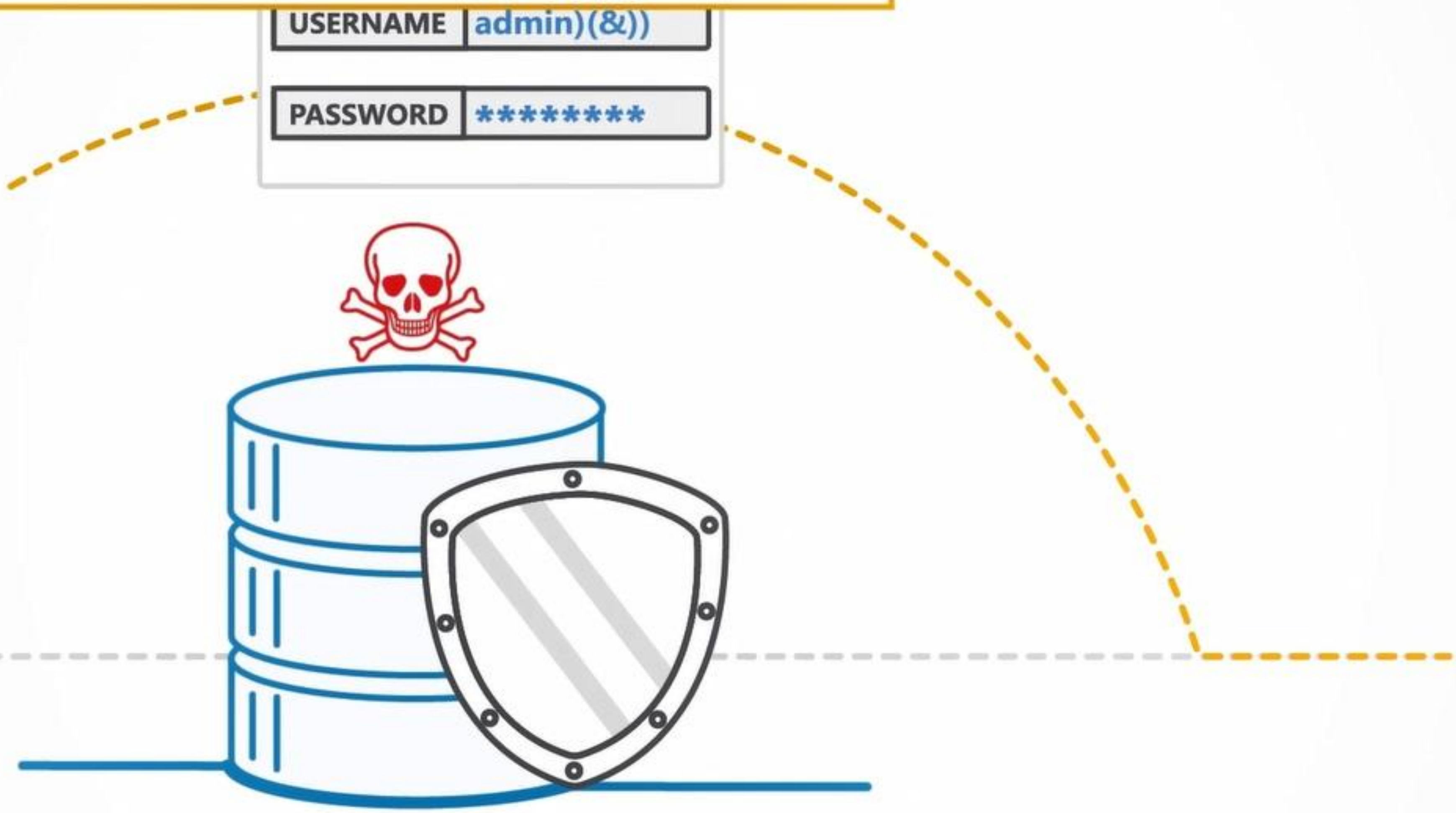
WHAT CAUSES AN LDAP INJECTION?



admin)(&))

****

If this input is not first validated, the LDAP query interpreter



USERNAME admin)(&))

PASSWORD *********

Access and Control

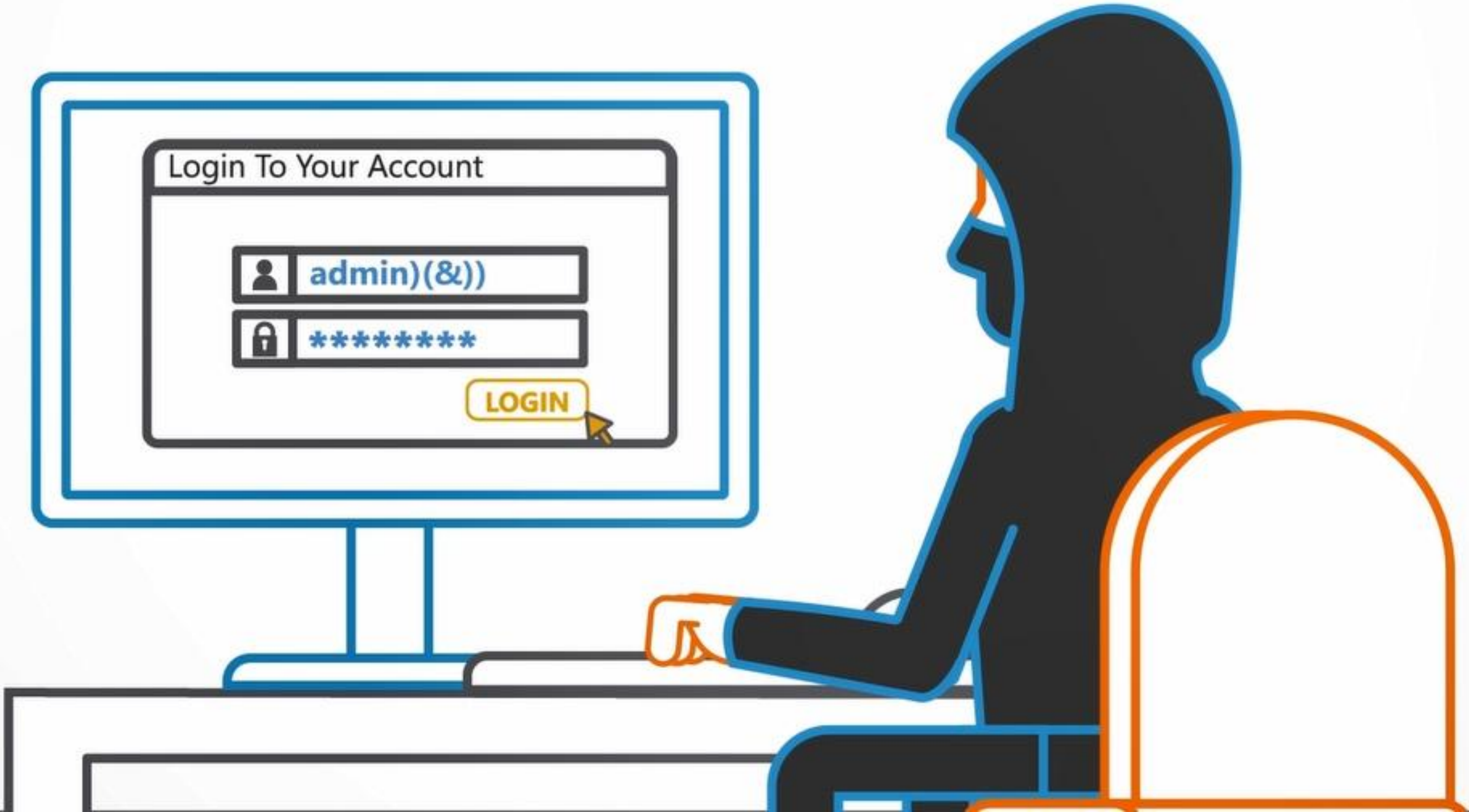
can be tricked into running arbitrary queries.

To understand

the LDAP Injection vulnerability,

let's look at an example of an Authentication bypass

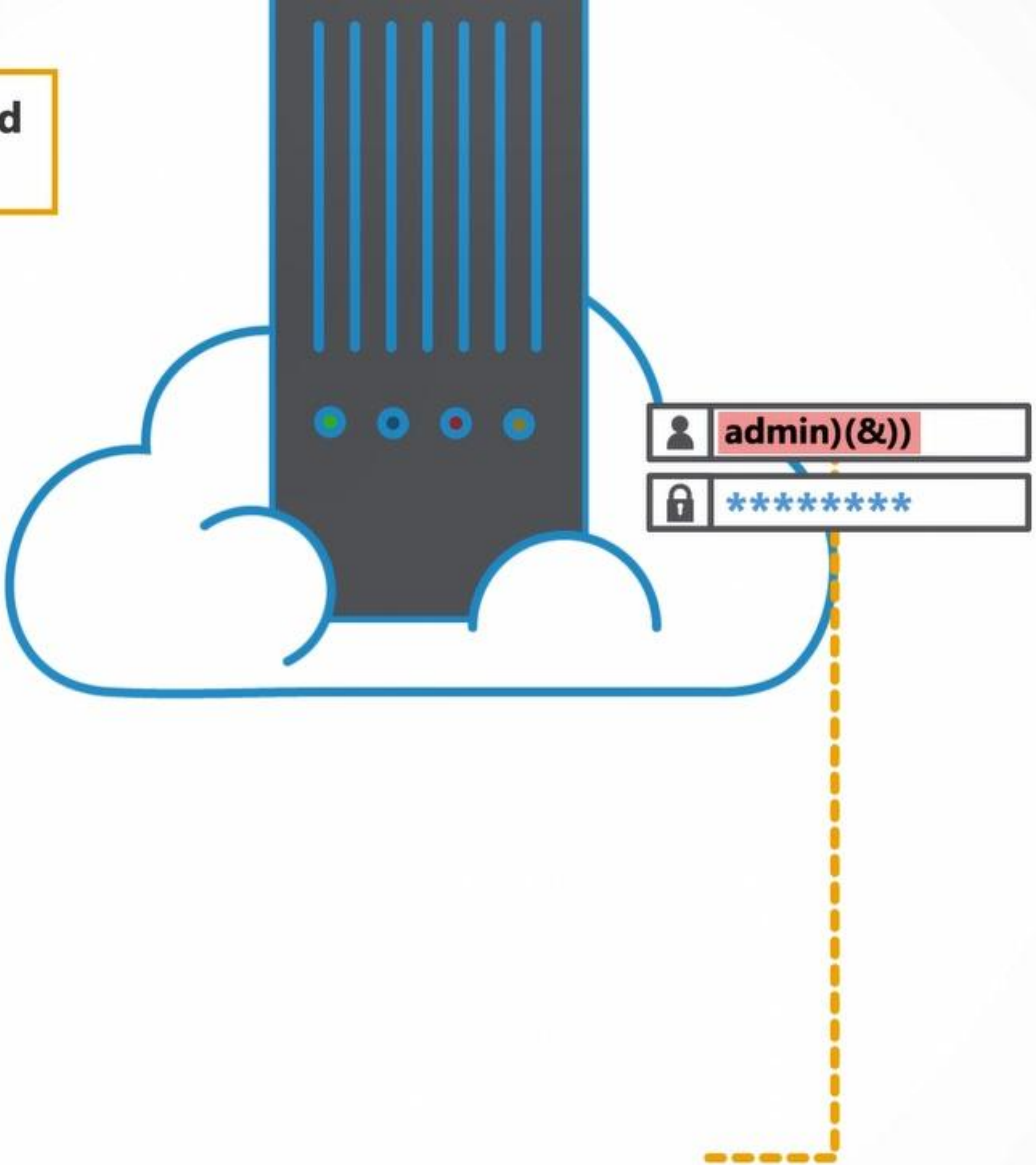
Here, an attacker submits input values that will take advantage



admin)(&))

****

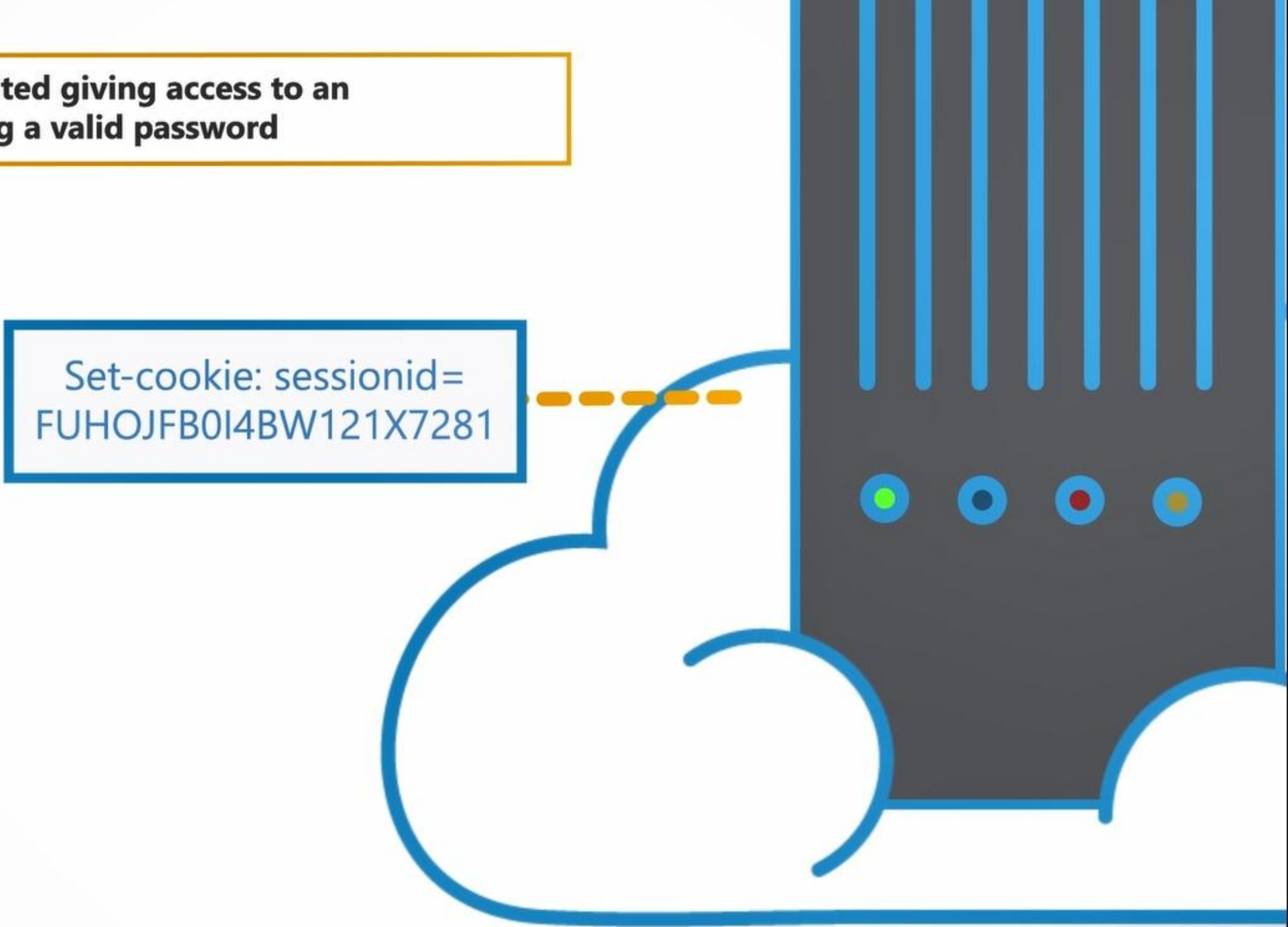
of a backend LDAP statement used to query users and passwords.



The submitted input changes the logic of the query. The ampersand in parentheses is interpreted as a "TRUE" statement. Because of this additional true statement - the password condition will be ignored

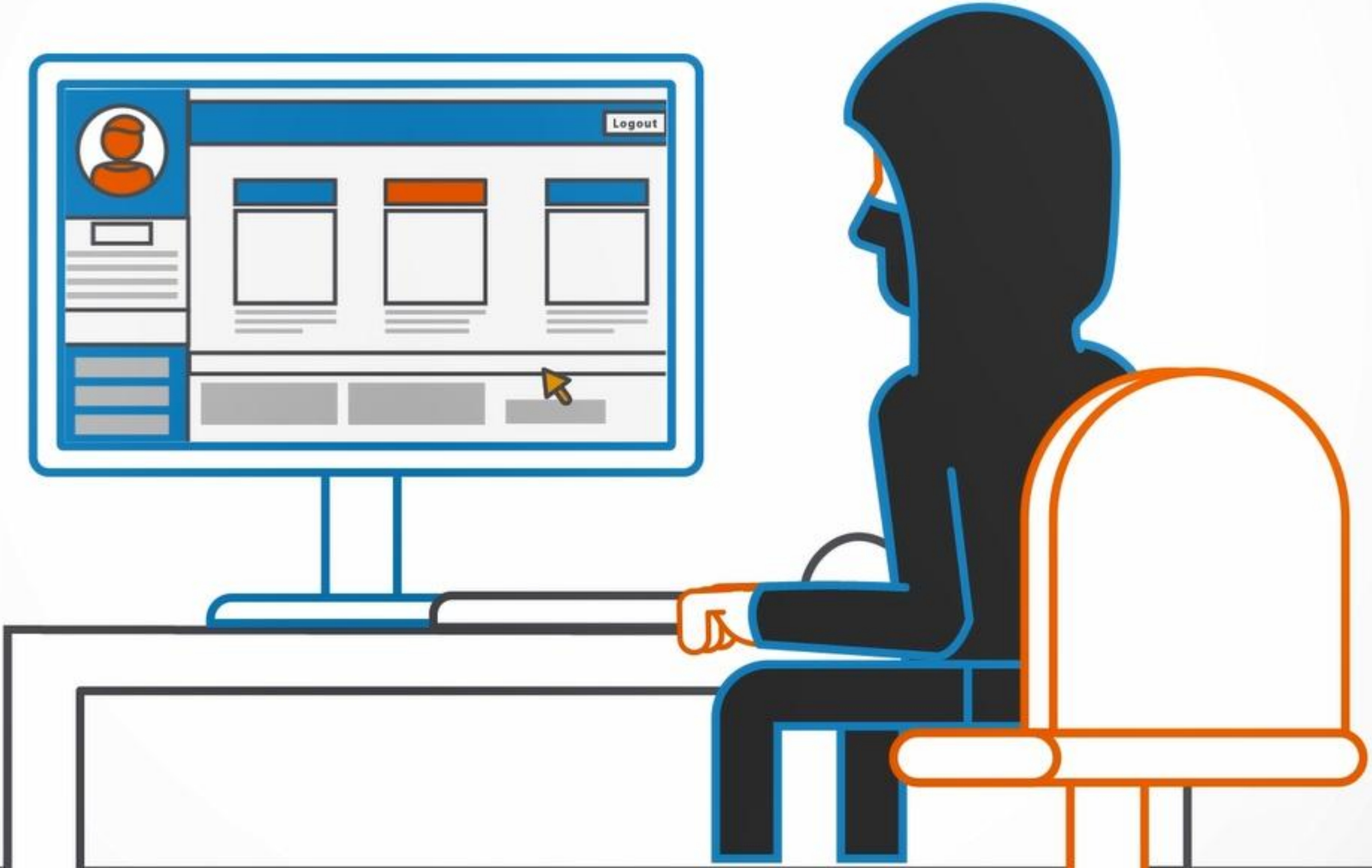
"(&) = TRUE"

The vulnerability is exploited giving access to an account without providing a valid password

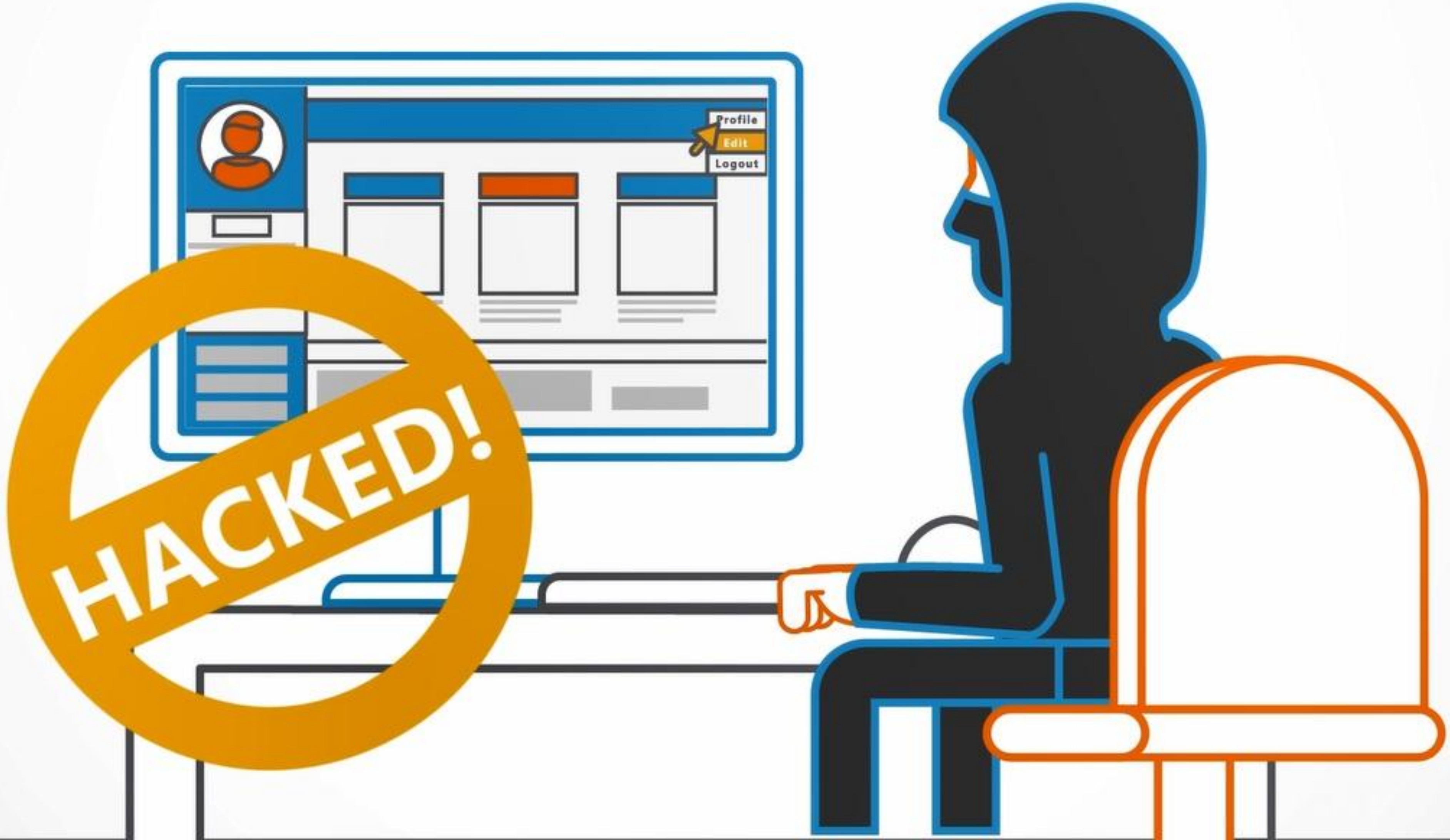


Set-cookie: sessionid=
FUHOJFB0I4BW121X7281

The session cookie is returned to the browser,



the attacker is now logged in as administrator.

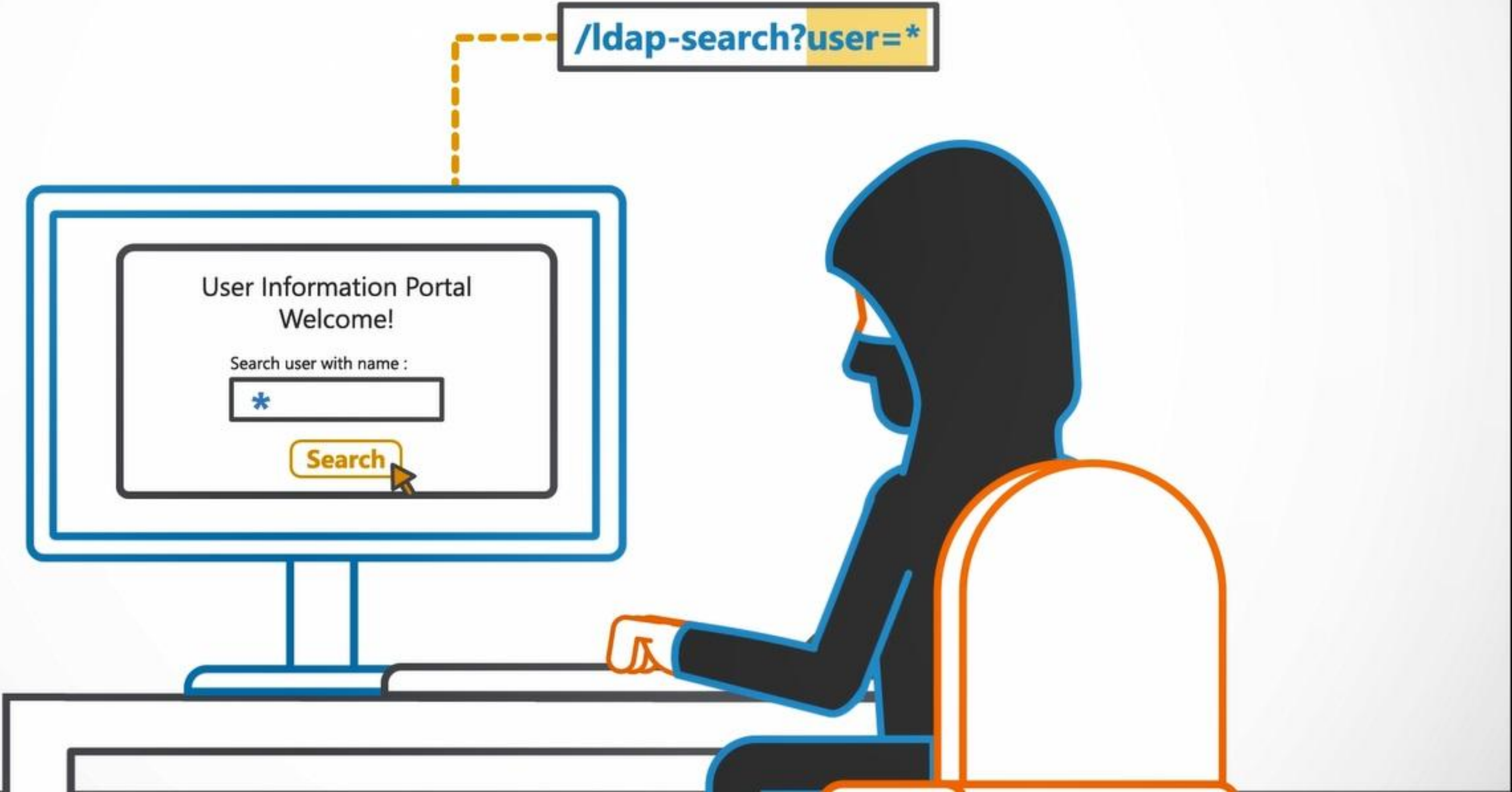


Let’s take a look at another example

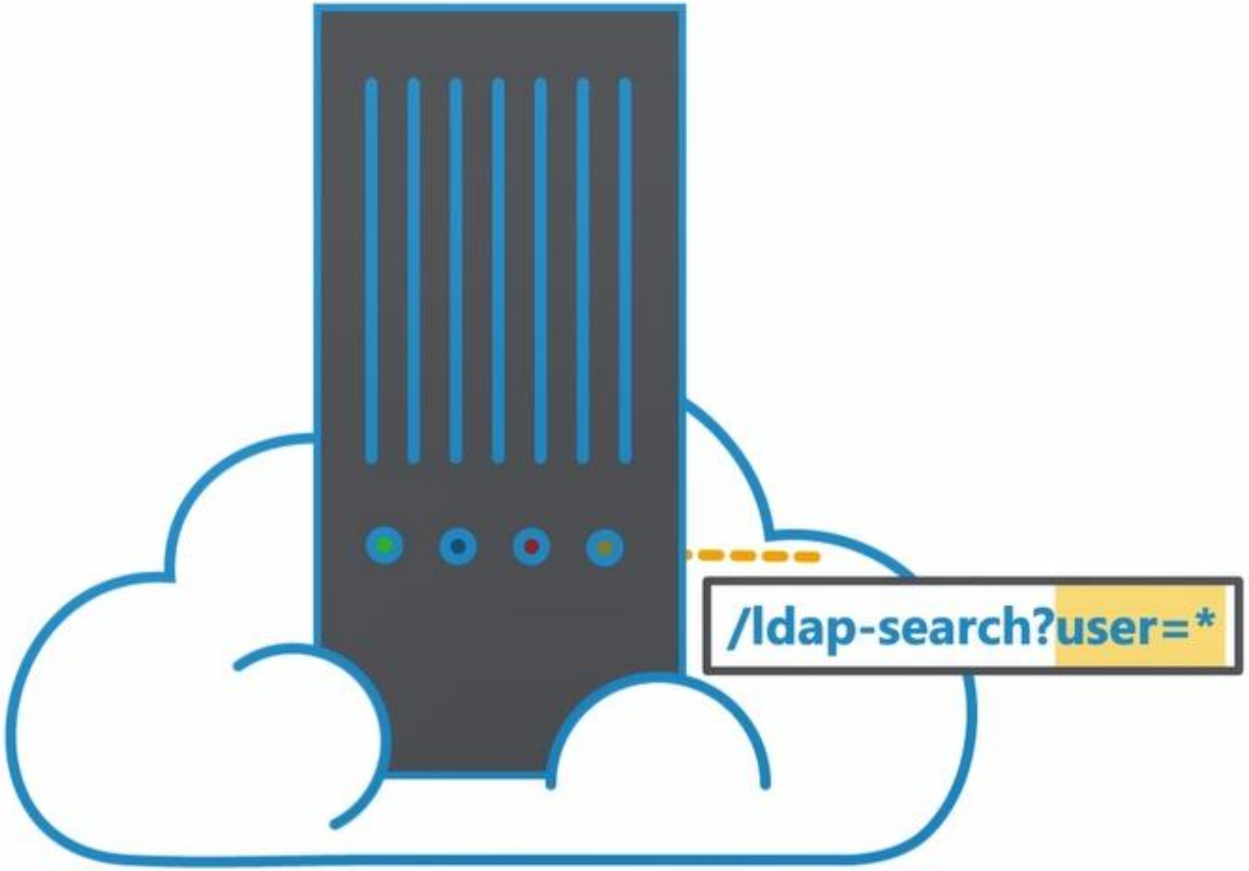
This time we’ll look at

Information Disclosure through an LDAP Injection

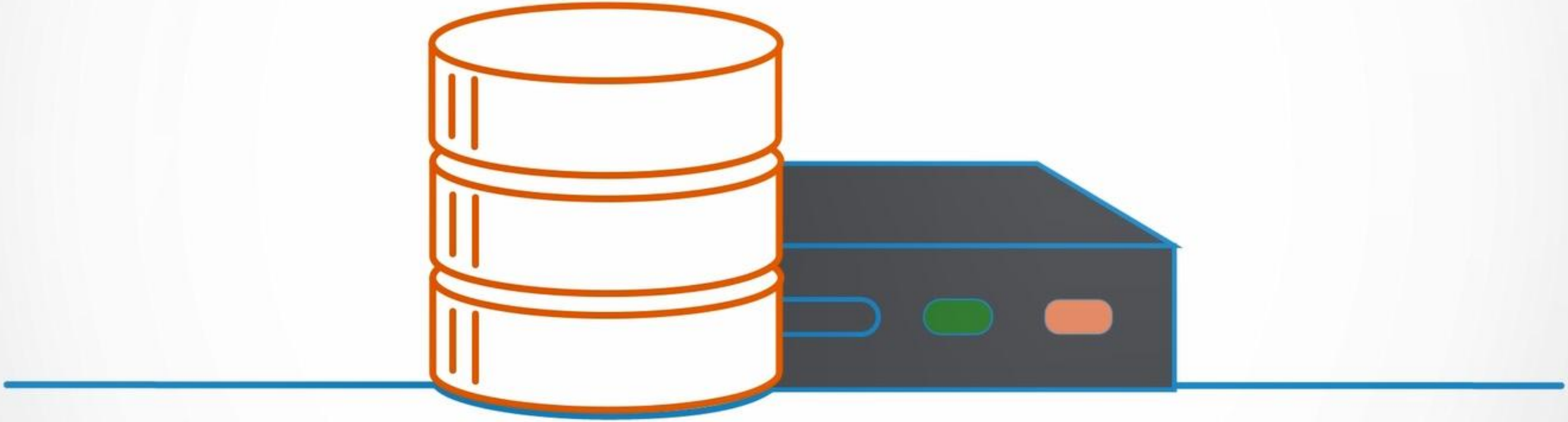
An attacker submits input values to take advantage of a different query...



The submitted input changes the logic of the query,

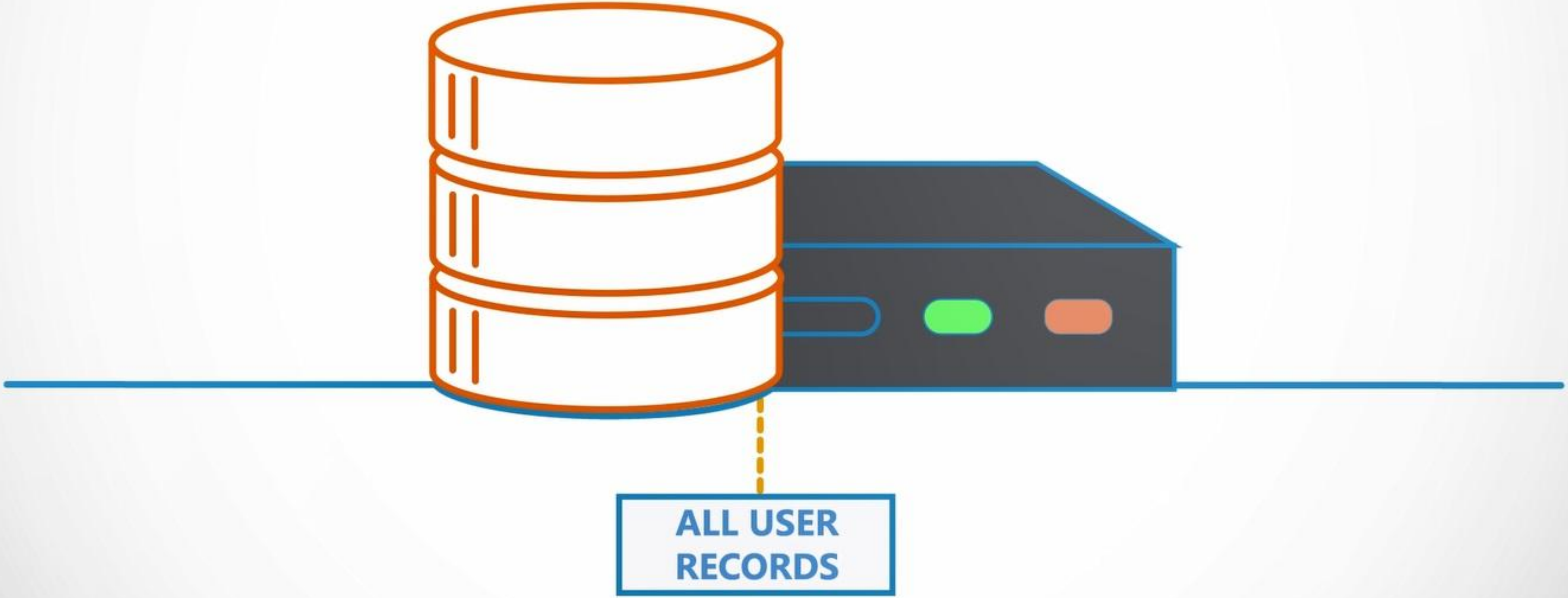


so that the wildcard statement queries



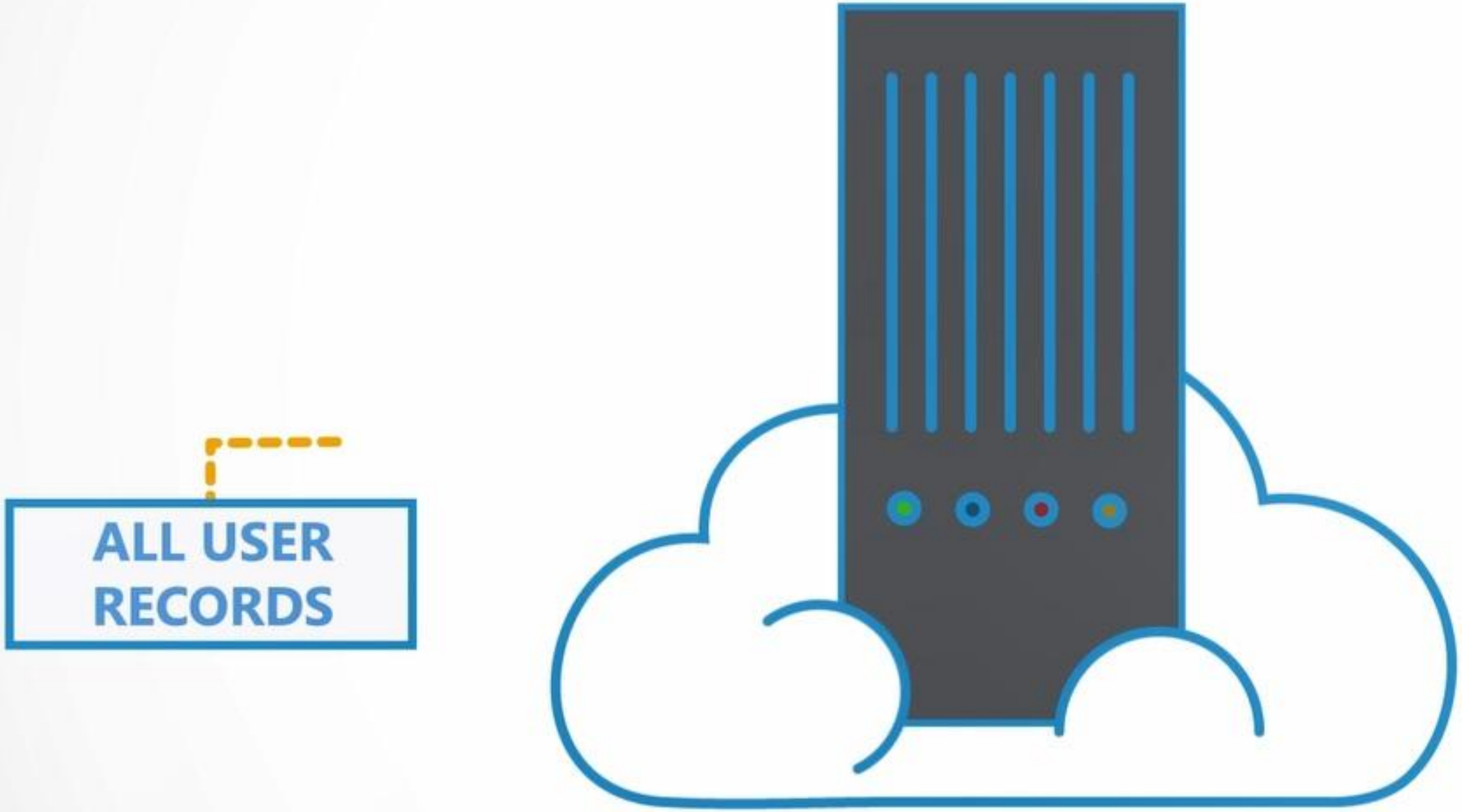
(uid=*)

all users in the LDAP tree.

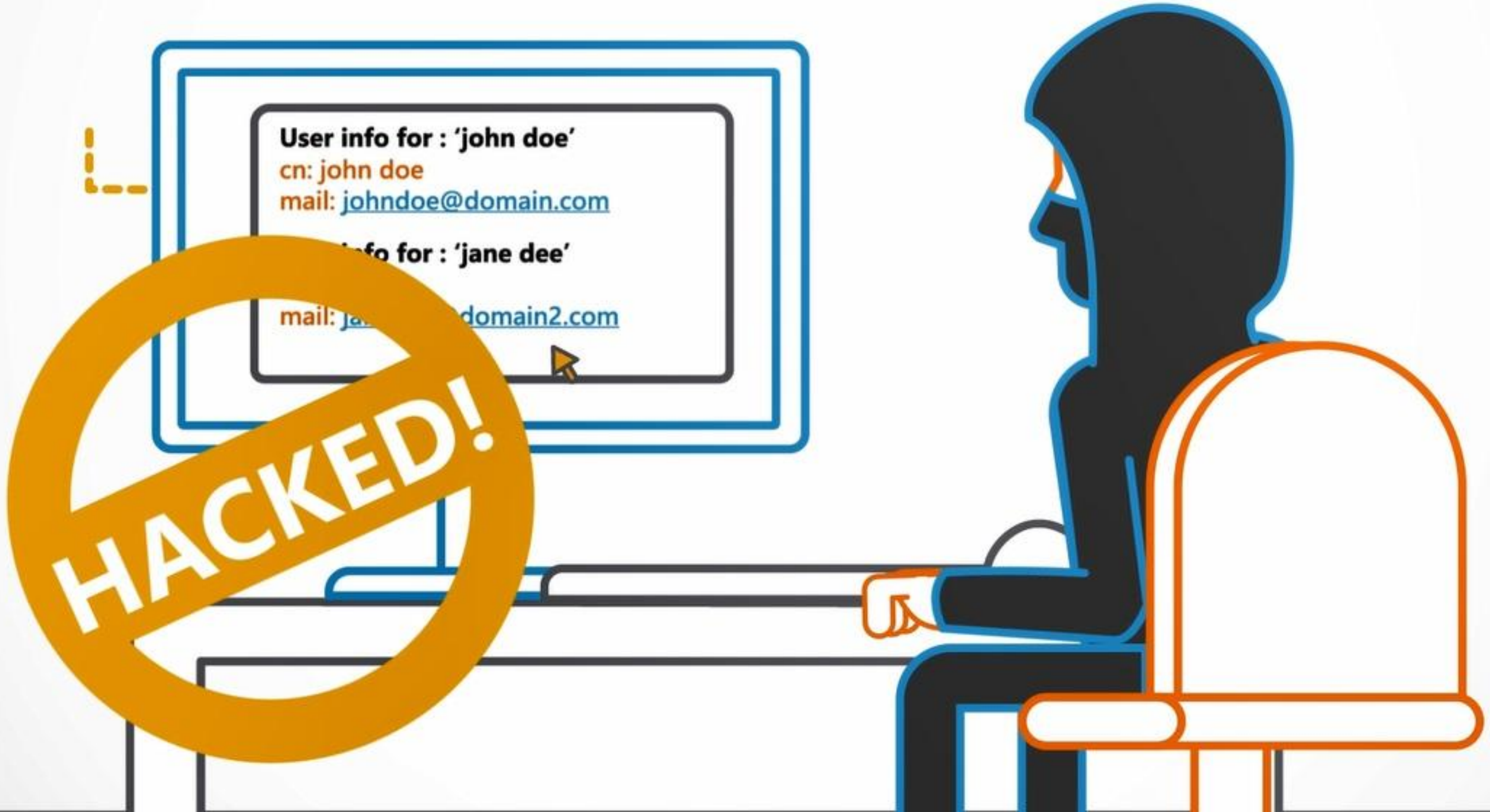


ALL USER RECORDS

The vulnerability is exploited in order to gain detailed information about all users in the LDAP tree,



resulting in a major breach of security.



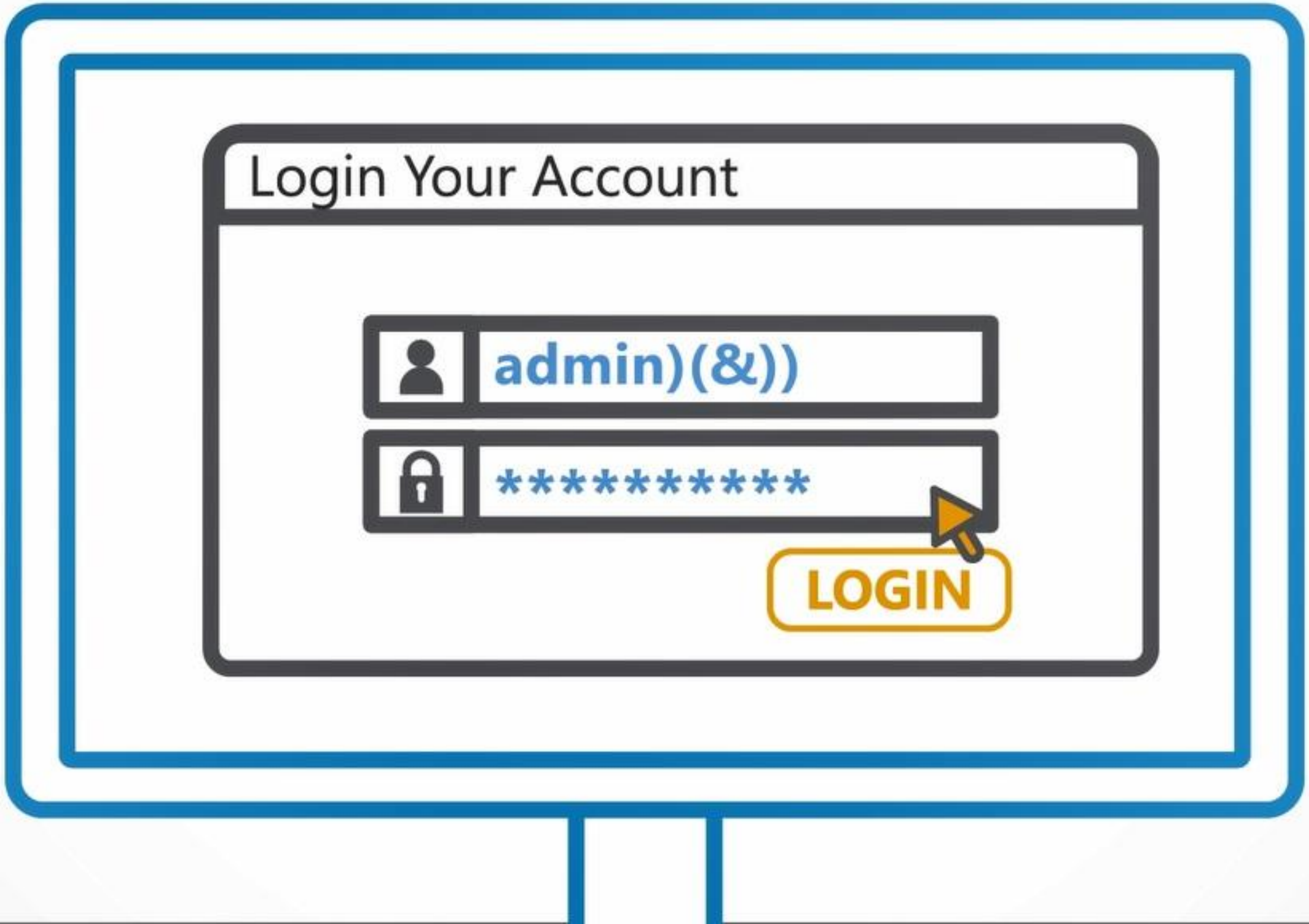
User info for : ‘john doe’
cn: john doe
mail: johndoe@domain.com

fo for : ‘jane dee’
mail: ja domain2.com

LDAP INJECTIONS CAN CAUSE SEVERE ISSUES

Sensitive information about users and hosts represented in the LDAP tree could be disclosed, modified, inserted or deleted.

LDAP injection could be used to bypass access control, and gain access to Administrator accounts.

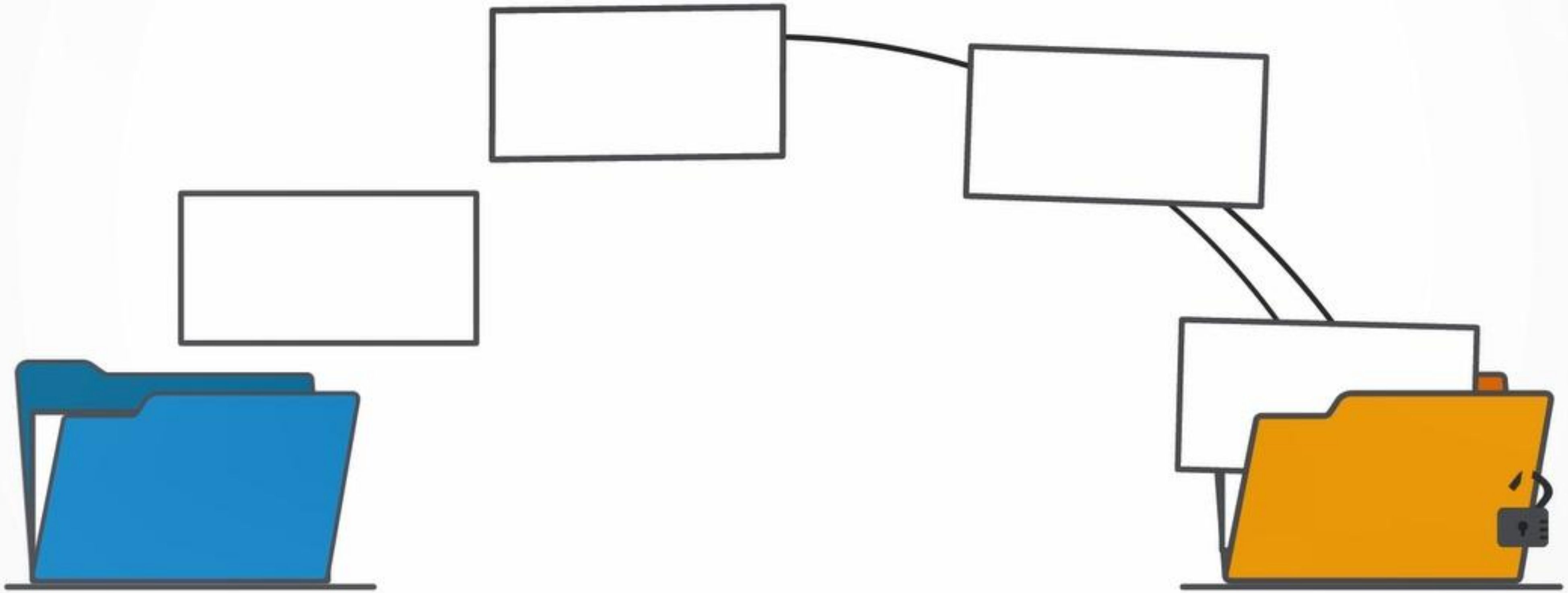


Login Your Account

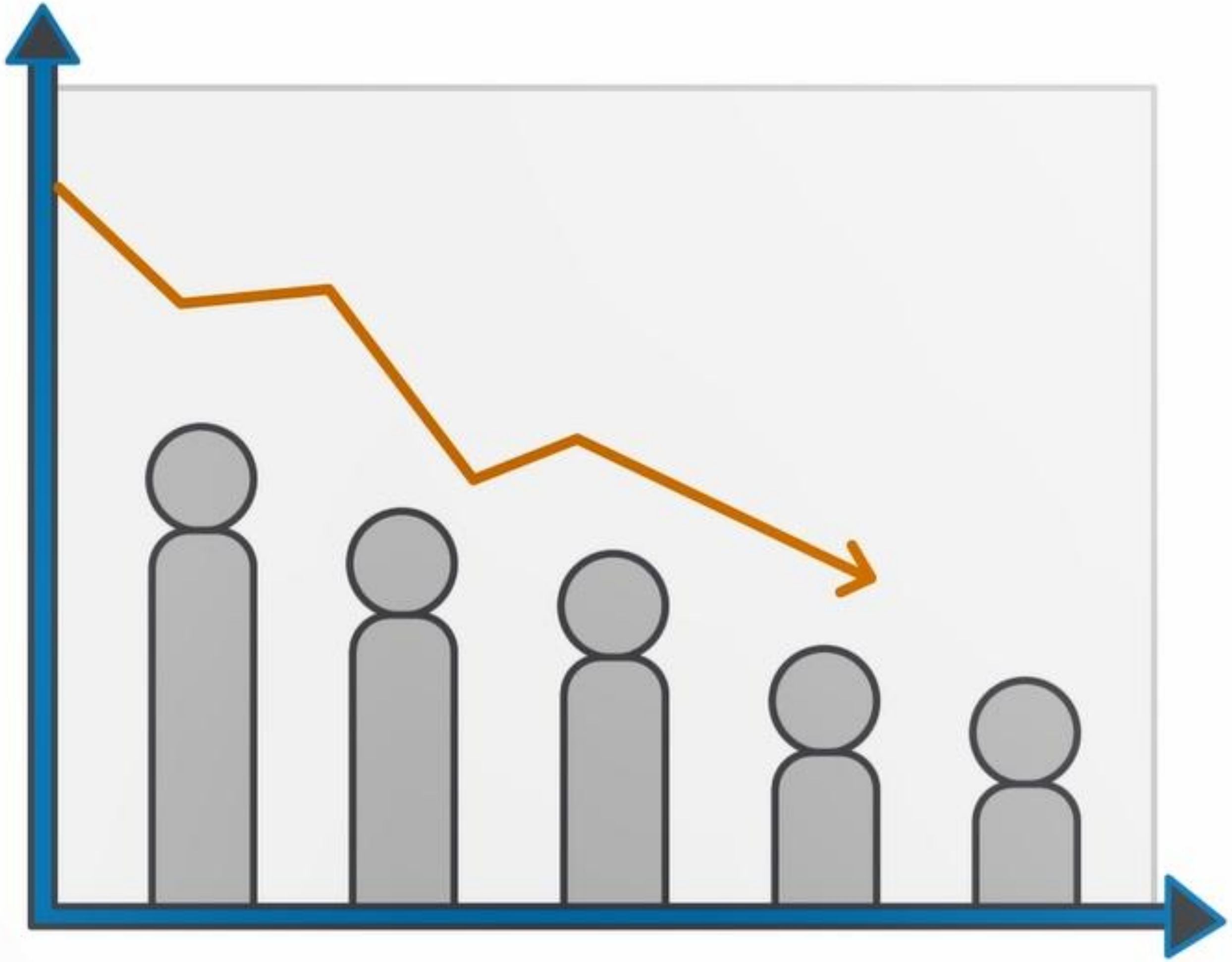
admin)(&))

****

Sensitive data could be exposed, leading to privacy issues.



This weakness can lead to full system compromise, loss of reputation and financial damages.



To prevent LDAP Injections

- User input that is being used as part of an LDAP query should be sanitized first. This includes GET and POST parameters, cookies and other HTTP headers

- Always use framework provided functions when available and make use of escaped variables in LDAP queries

To prevent LDAP Injections

- Use LDAP Injection resistant frameworks, automatic LDAP encoding, and framework provided functions, where possible
- Also, perform validation through an allowlist
- Minimize LDAP binding account privileges by using the Least Privilege principle

Congratulations,

you have now completed this module, LDAP Injections!

SECURE CODE

WARRIOR

www.securecodewarrior.com