SECURE CODE

WARRIOR

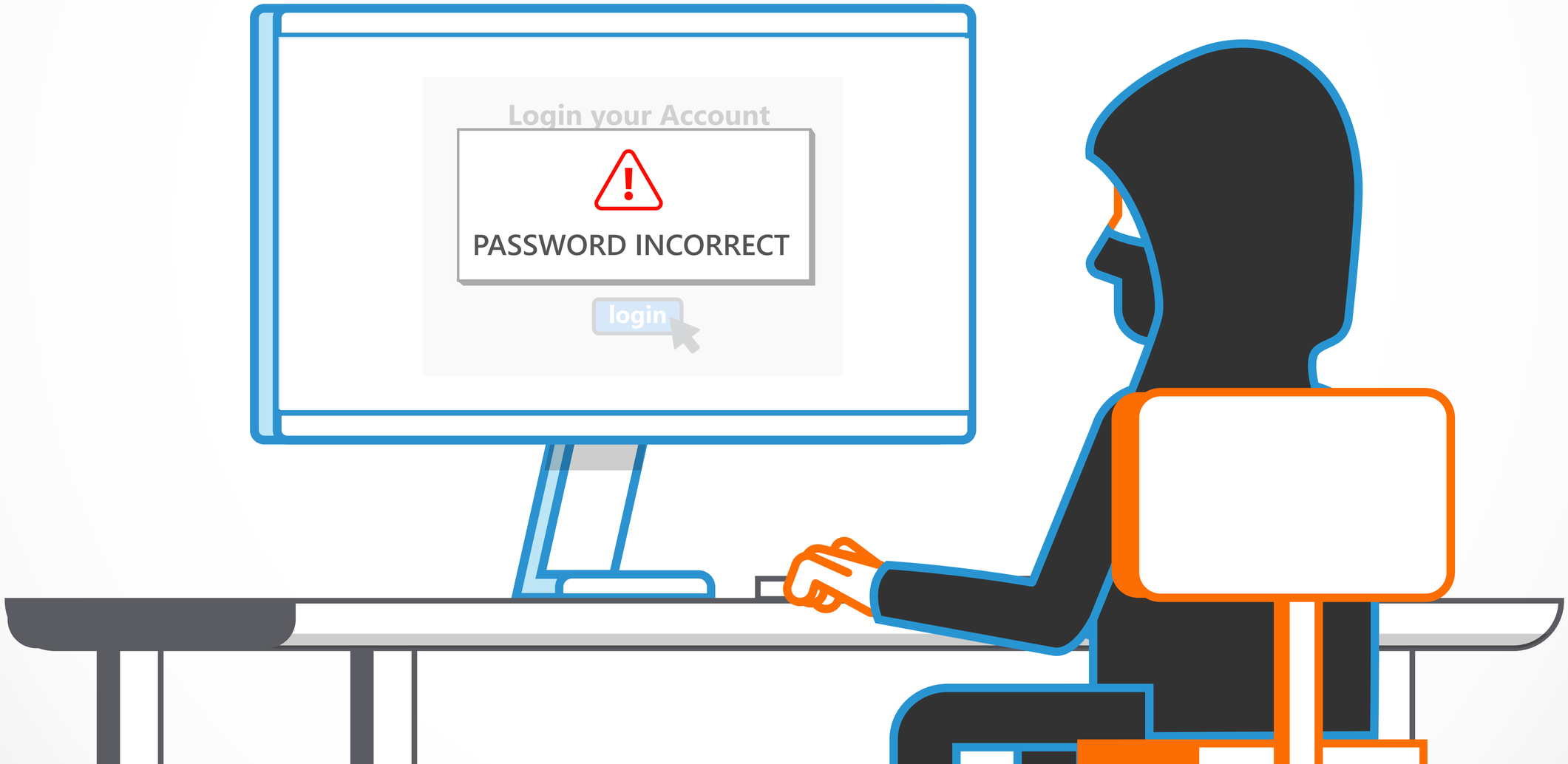
ROBUST ERROR CHECKING

We'll discuss

Robust Error Checking and
the associated hazards

SO, WHAT IS “ROBUST ERROR CHECKING”?

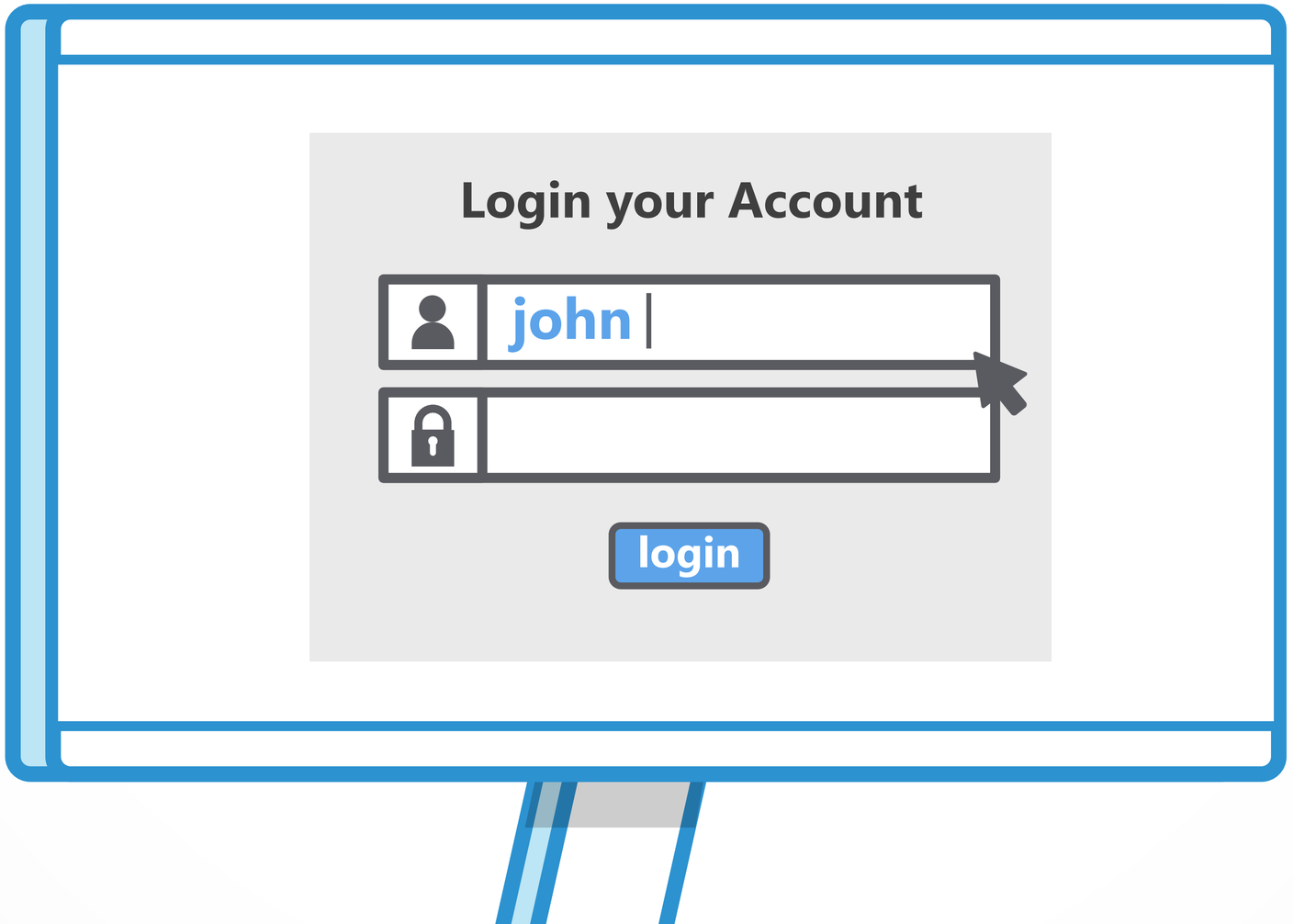
This is the process of ensuring that sensitive data such as stack traces or information about infrastructure, is not leaked through error messages.



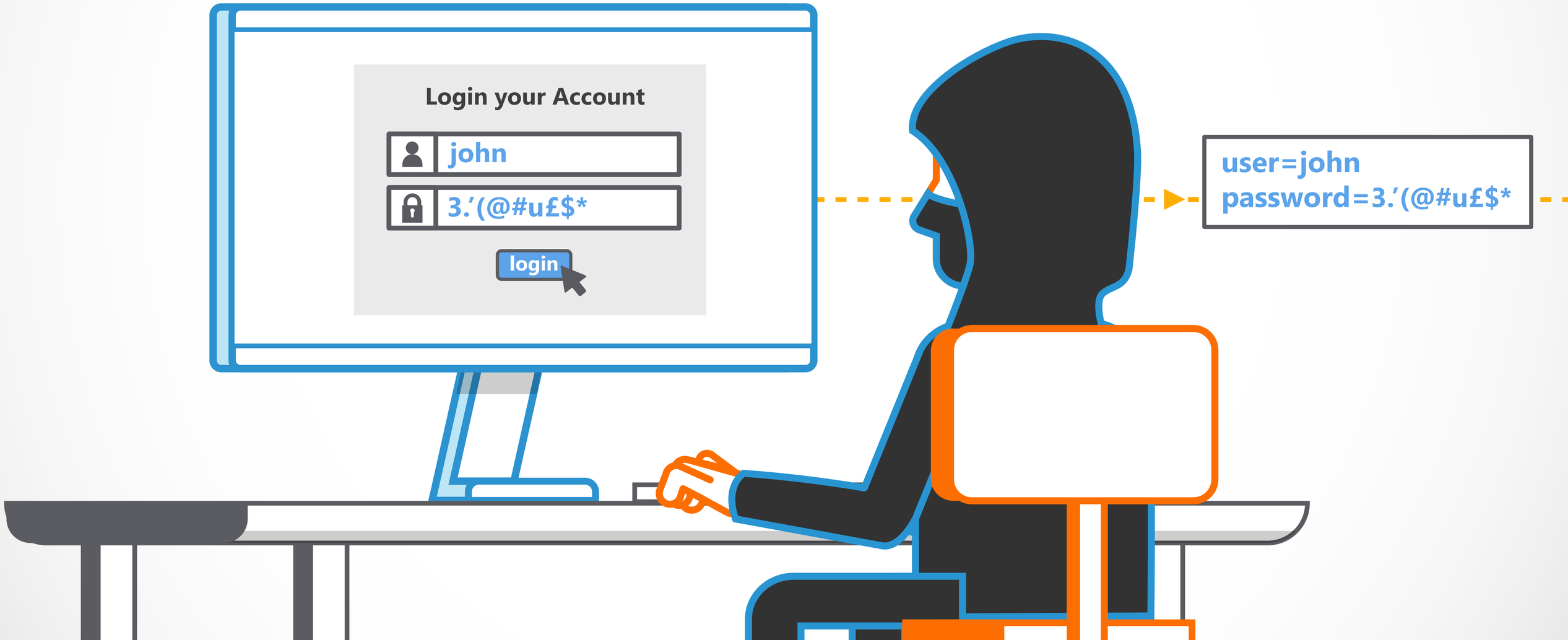
LET’S LOOK AT AN EXAMPLE.

An adversary attacks an application where exception handling has not been implemented correctly.

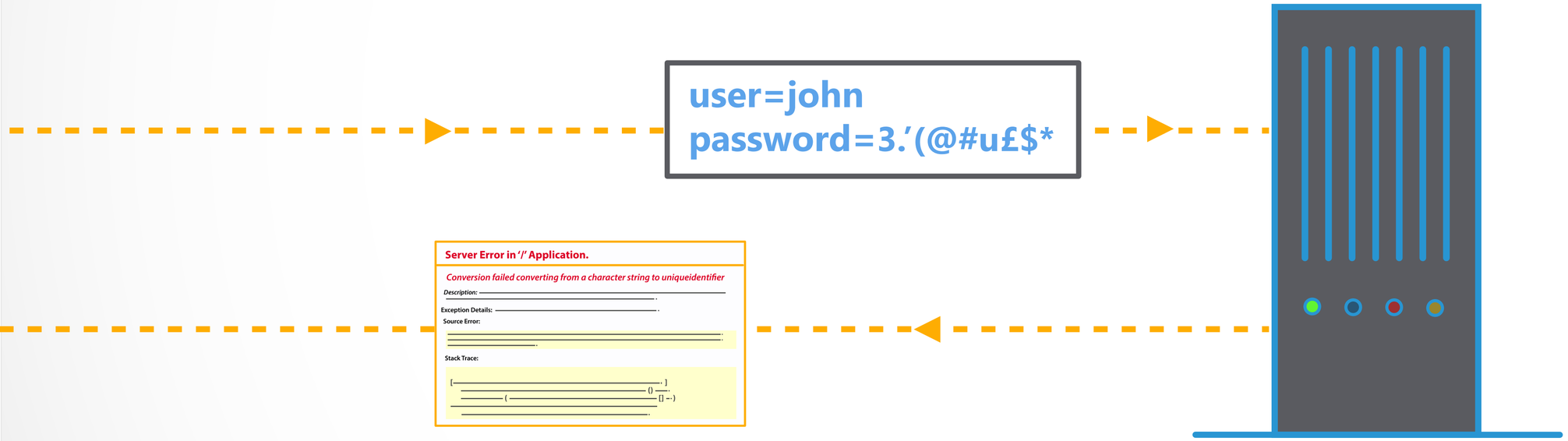
UNHANDLED EXCEPTIONS



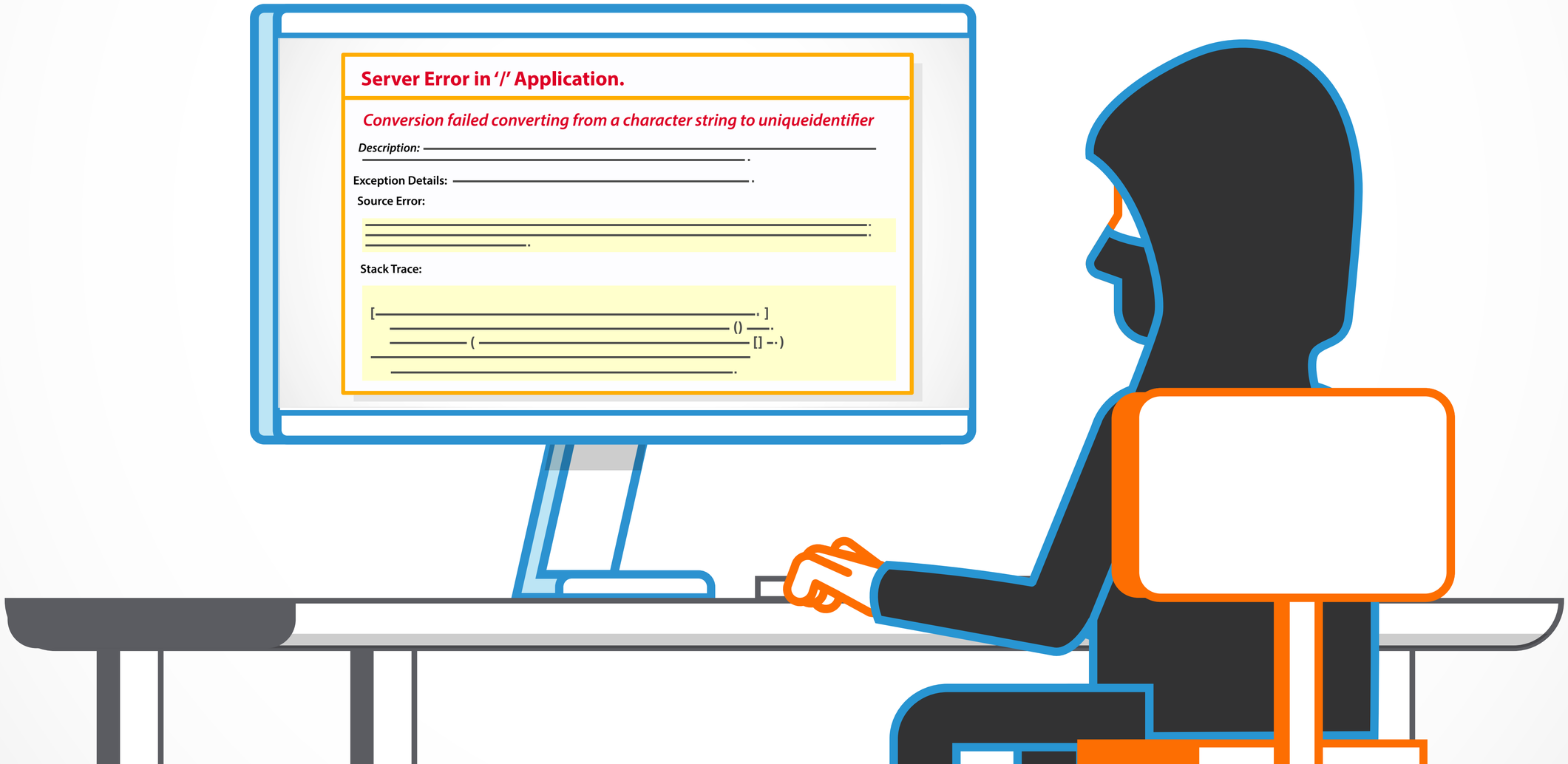
By providing unexpected data into an input field, the attacker is able to force an error from the application.



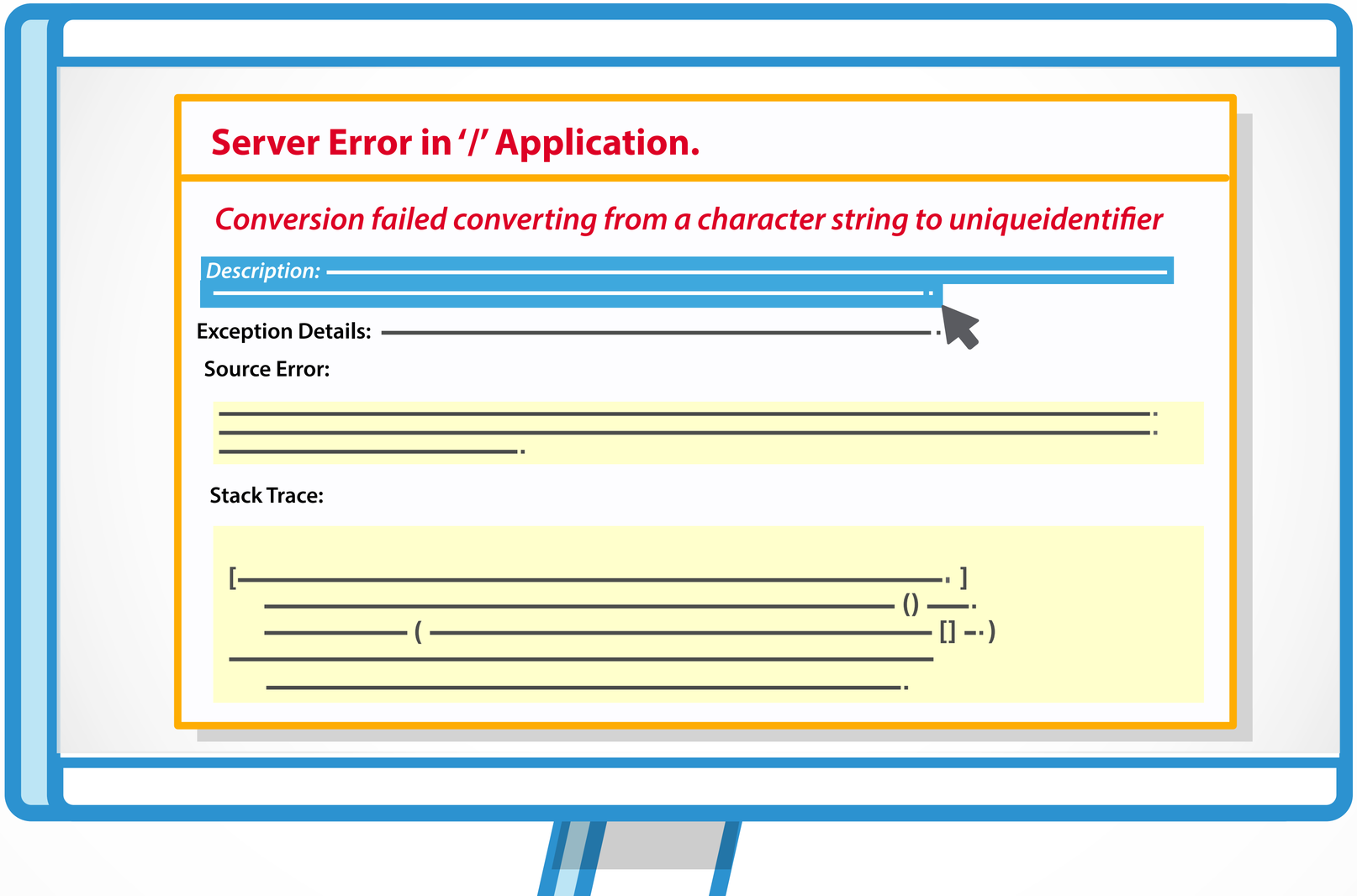
An error page with stack trace information is returned.



The attacker now knows more intel about the application.



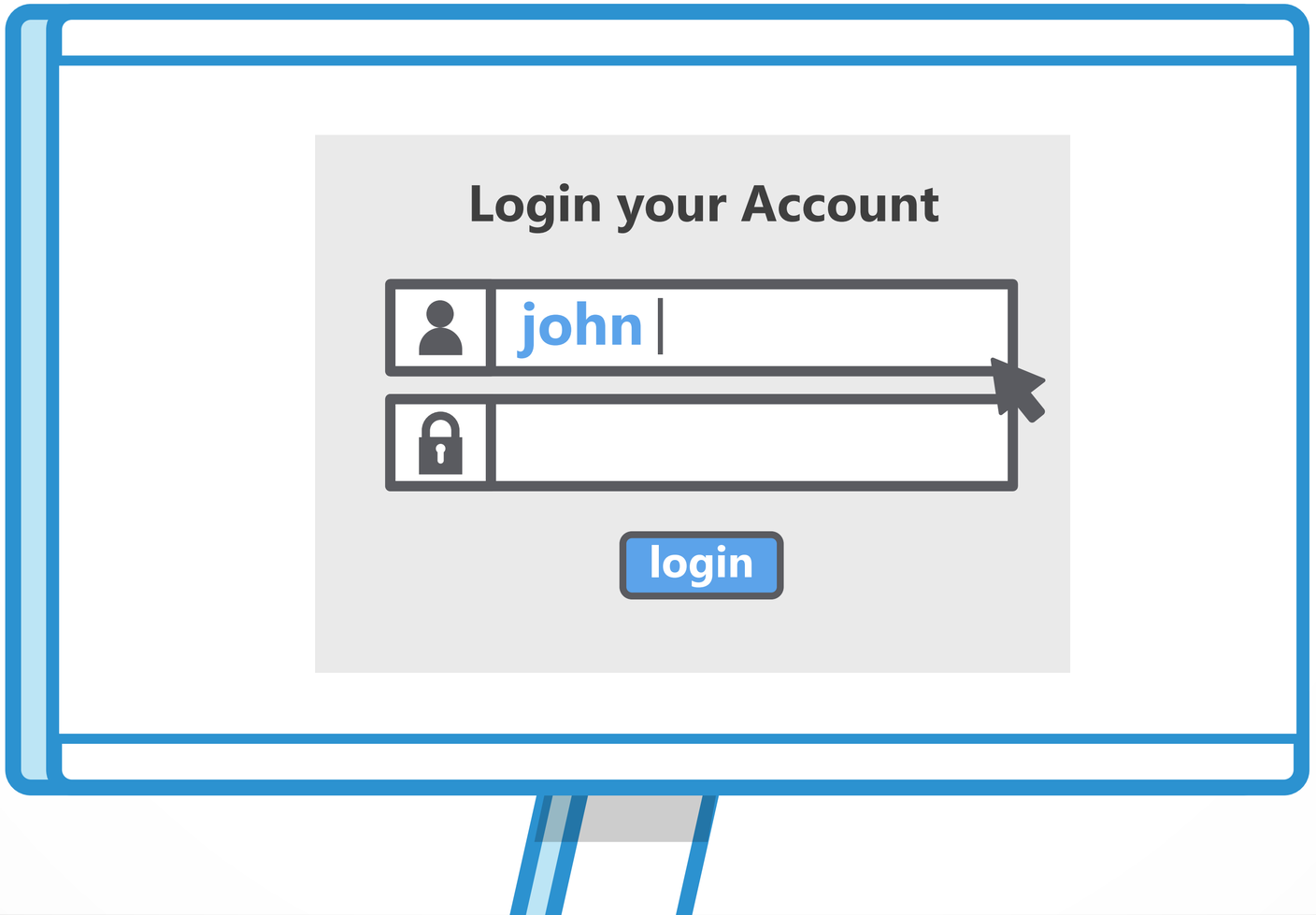
He’s been provided information on what frameworks are used, what database connections are made and which part of the code broke the application.



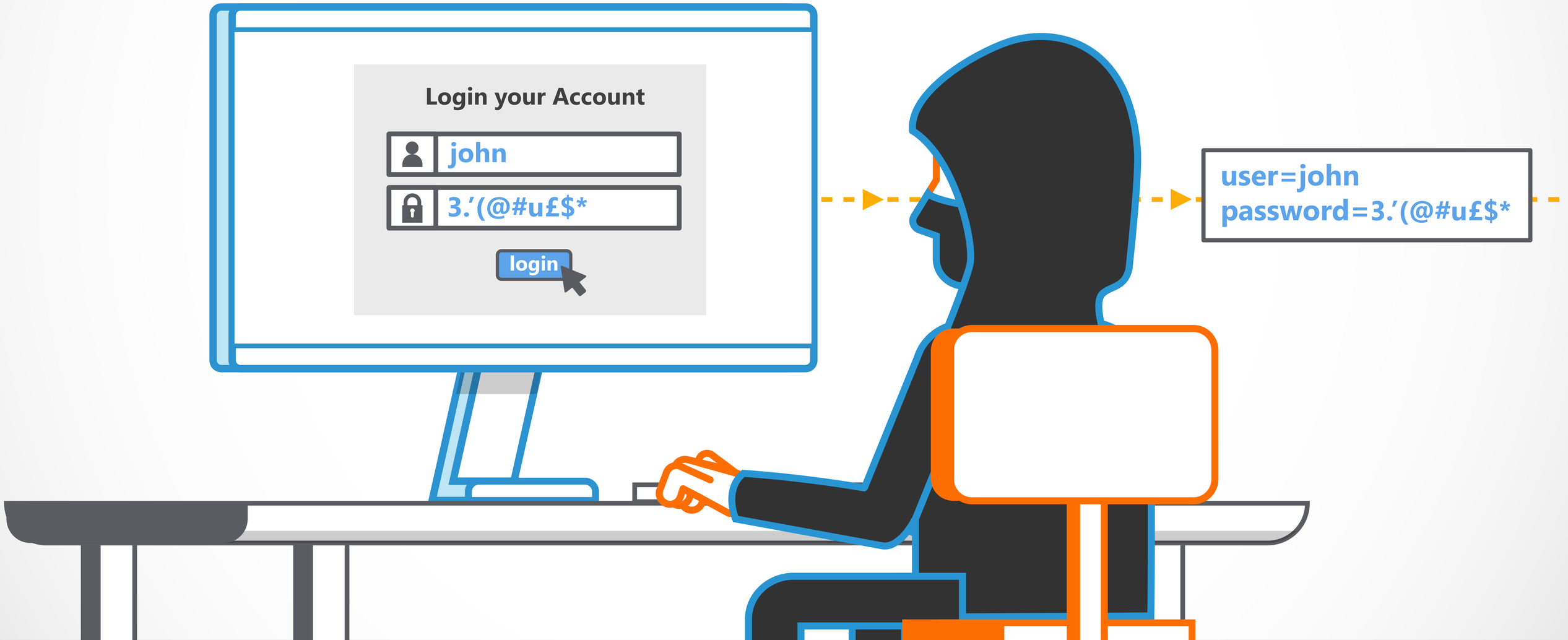
AN APPLICATION WHICH HAS *CORRECTLY* IMPLEMENTED ROBUST ERROR CHECKING.

Here, an application has implemented robust error checking in a way that handles errors properly without providing too much information.

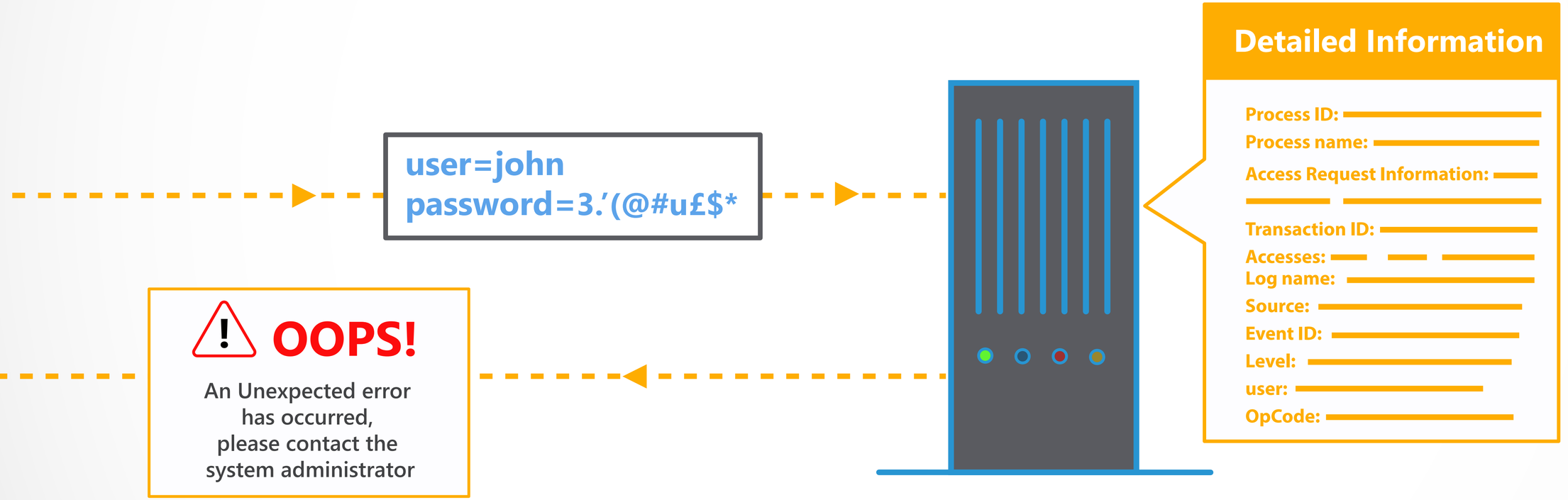
PROPERLY HANDLING EXCEPTIONS



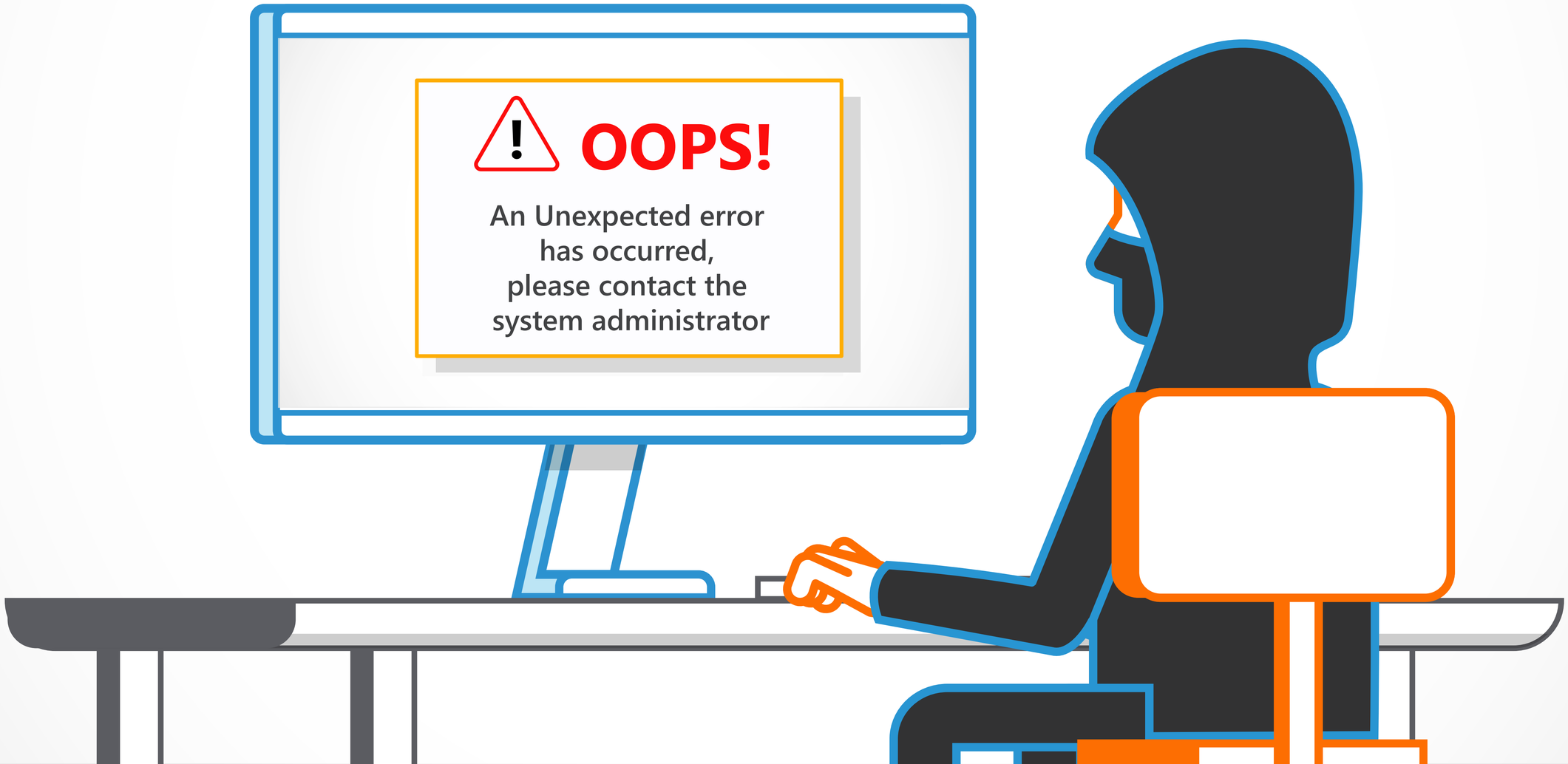
An attacker tries breaking the application by providing unexpected data to an input field hereby forcing an error.



A generic error message is provided to the end-user. Detailed information is logged on the server, so the administrator can investigate.

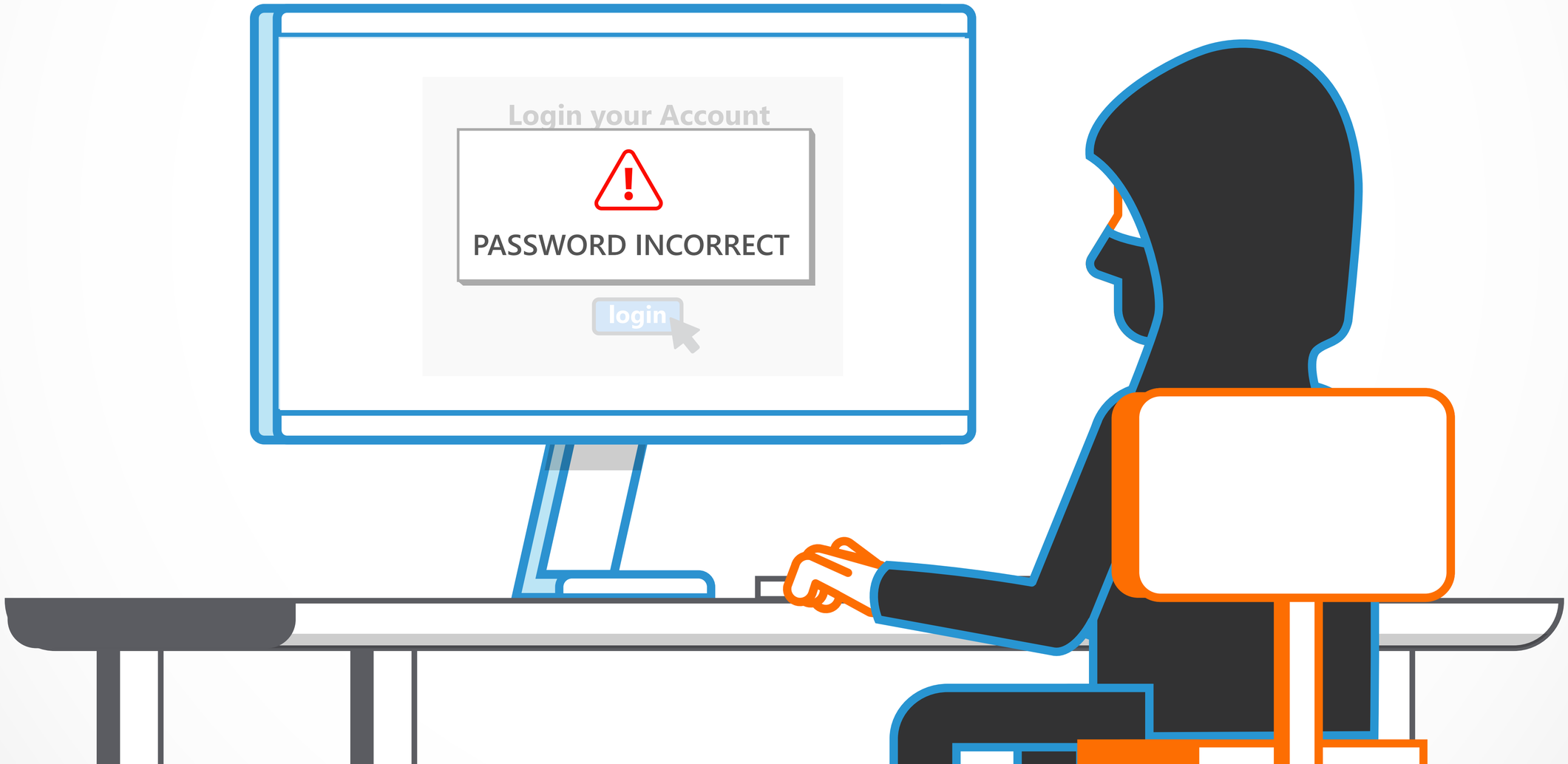


Because no information can be inferred from the error message, the attacker can not infer anything about the applications internals or what caused the error.

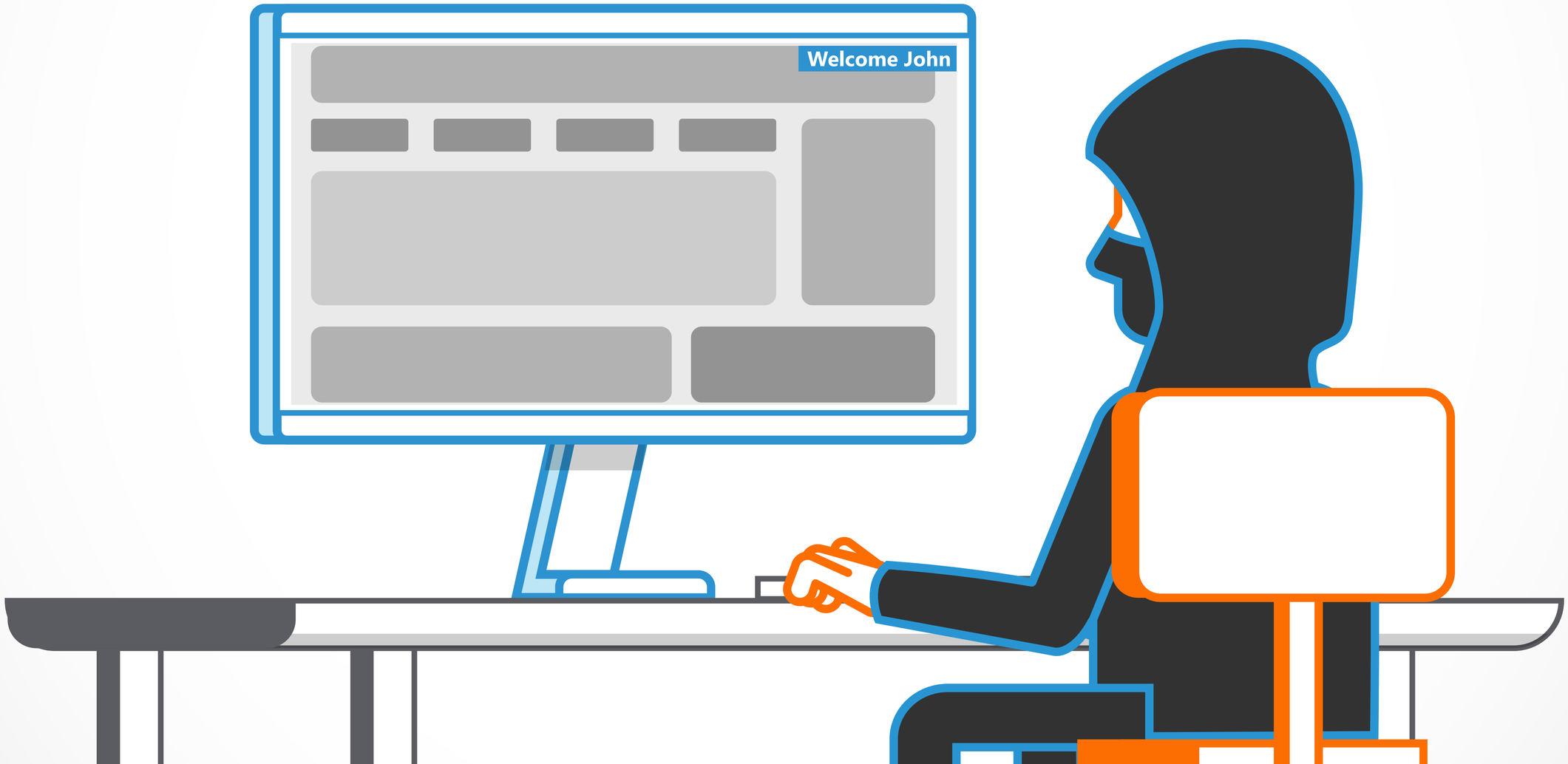


WITHOUT ROBUST ERROR CHECKING,
YOUR APPLICATIONS WILL BE VULNERABLE.

An attacker could be provided with sensitive information that could help identify weak spots in the application.



These weak spots could then be investigated by savvy adversaries to leverage attacks against the application.



Although the "Robust Error Checking" concept will not stop attackers, it will make it much harder for them to analyze the inner workings of the system.

- So, when an unexpected error occurs, users should be presented with as little information as possible.
  Use generic error messages. Furthermore, the application should close in a controlled and secure way.

- Do not disclose private information.
  This means no stack traces, internal IP user information or library information should be displayed to a user.

- As a rule, you should write error information to a log file for further analysis by an administrator.

- Finally, make sure the system catches all possible errors to ensure the continuation of the application.

Congratulations, you have now completed this module, Robust Error Checking!

SECURE CODE
WARRIOR

www.securecodewarrior.com