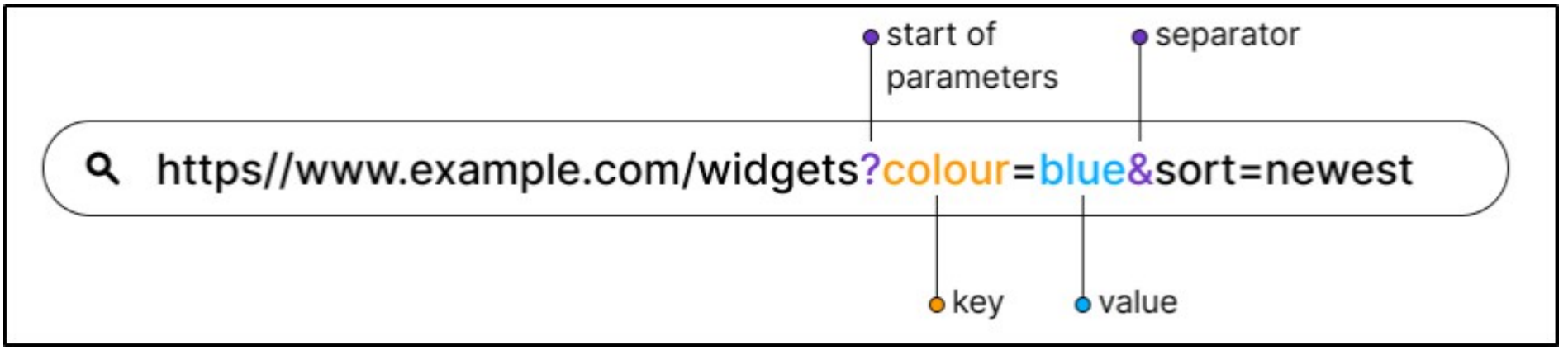
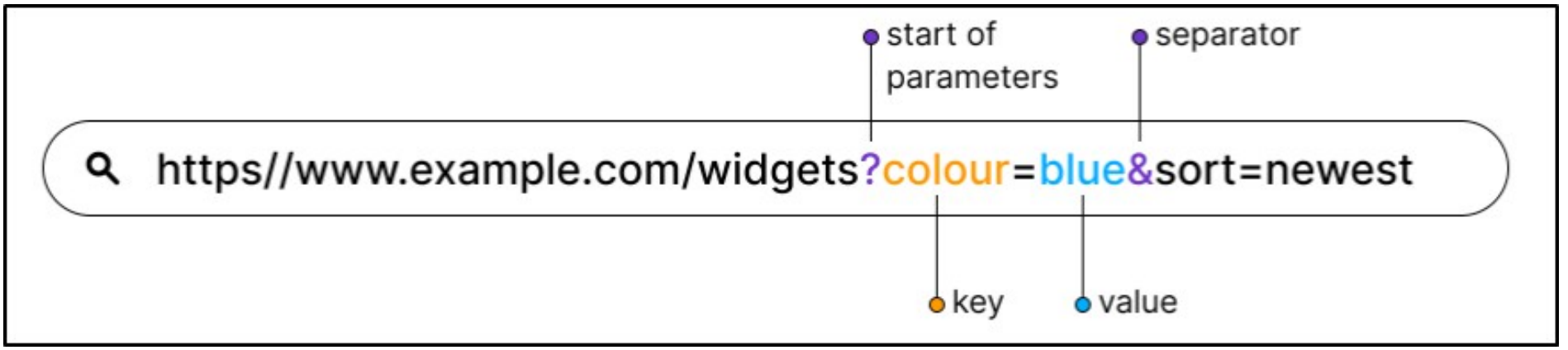
URL Parameters



URL parameters are variables attached to the end of URLs. They can be identified by the ? (question mark) directly after the webpage or directory name, followed by the parameter key name, then the = (equals) sign, then the value.

URL Parameters



If there are multiple parameters included in the same URL, then they are separated by the & (ampersand) symbol.

URL Parameters Use Cases

There are a few different reasons why webpages use URL parameters. The most common one is for search queries.

URL Parameters Use Cases

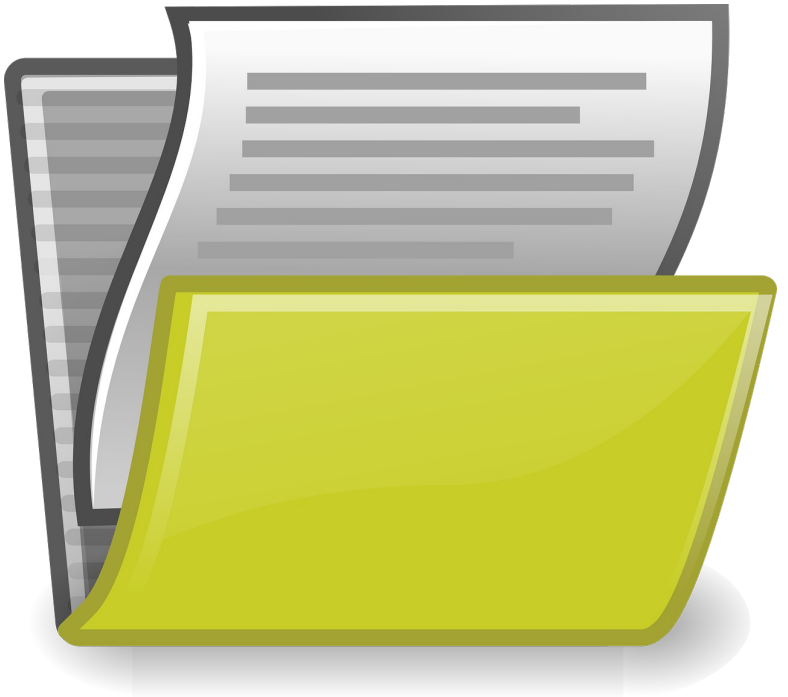
However, another common, and potentially dangerous use of URL parameters is to instruct the webserver on which webpage to display.

URL Parameters Use Cases

The use of URL parameters which reference other files on the webserver could potentially be exploited in an attack called Local File Inclusion (LFI).

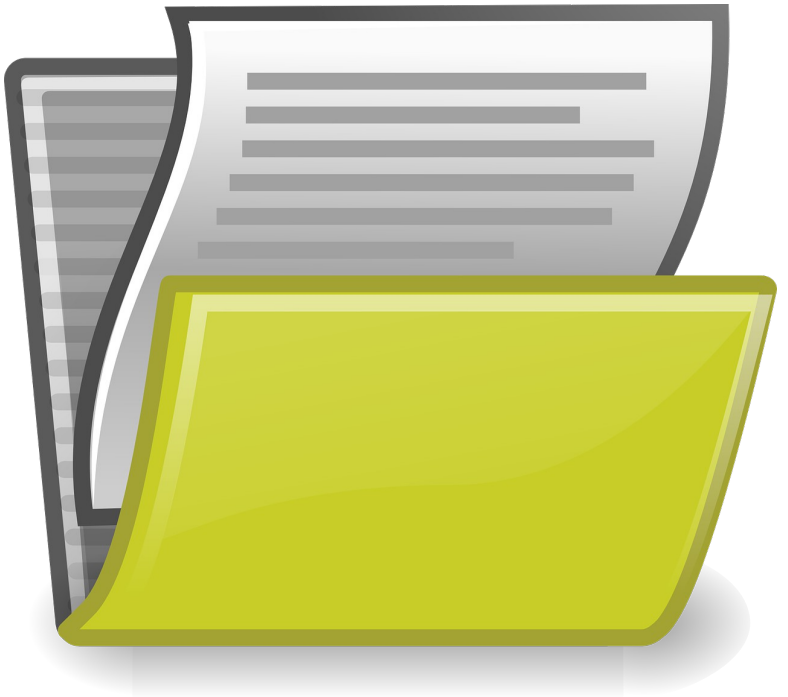
Local File Inclusion

Local File Inclusion (LFI) is a web app vulnerability where arbitrary local webserver files can be accessed through a web interface.



Local File Inclusion

LFI vulnerabilities can lead to sensitive data exposure, and can also be used as the first step in a chain of exploits.



Local File Inclusion

The inclusion of file names in URL parameters is a typical method through which a potential LFI vulnerability is identified.

Local File Inclusion: Filesystem Structure

Each **../** indicates an elevation of one level in the filesystem, traveling from the web app's working directory up to the top-level directory (/)

/
/var/
/var/html/
/var/html/www/

Local File Inclusion: Filesystem Structure

From the top-level directory, we can provide a filepath to the file we want to access.

A typical test file for LFI on Linux / Unix webservers is the **/etc/passwd** file, since it is publicly readable by default, and gives info regarding usernames on the webserver.

Local File Inclusion: User Enumueration

```
admin:x:1000:1000:admin,,,:/home/admin:/bin/bash
```

Using LFI, we are able to view the **/etc/passwd** file, which includes a user named **admin**. If the webserver is not secured properly, we could view this user's common private files.

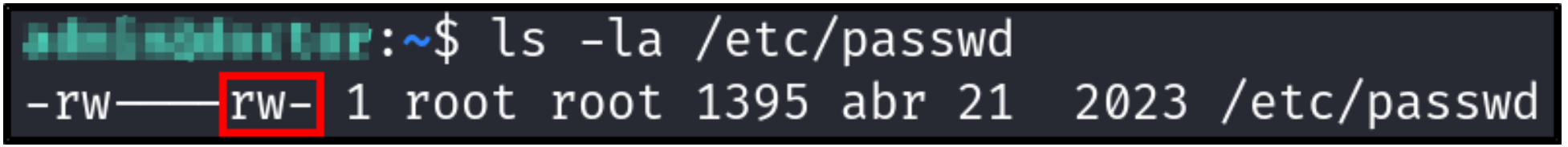
Local File Inclusion:
SSH Private Key Capture

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,9FB14B3F3D04E90E

uuQm2CFIe/eZT5pNyQ6+K1Uap/FYWcsEklzONt+x4AO6FmjFmR8RUpwMHurmbRC6
hqyoiv8vgpQgQRPYMzJ3QgS9kUCGdgC5+cXlNCST/GKQOS4QMQMUTacjZZ8EJzoe
o7+7tCB8Zk/sW7b8c3m4Cz0CmE5mut8ZyuTnB0SAlGAQfZjqsldugHjZ1t17mldb
+gzWGBUmKTOLO/gcuAZC+Tj+BoGkb2gneiMA85oJX6y/dqq4Ir10Qom+0tOFsuot
```

Such as a user's SSH private key file, which can be captured and used to login to the server as that user.

Privilege Escalation
Passwd Write Permission Abuse



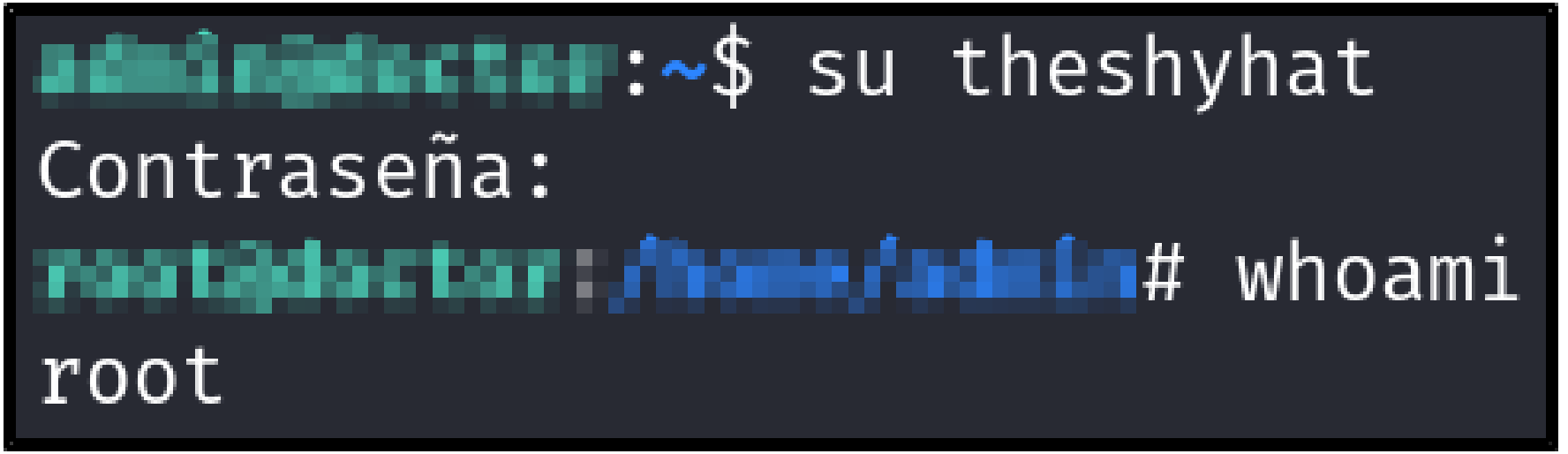
This system's **passwd** file has insecure permissions, allowing any user to modify it.

Privilege Escalation
Passwd Write Permission Abuse

```
echo 'theshyhat:HYvTBkaCsoqLA:0:0:root:/root:/bin/bash' >> /etc/passwd
```

The second step is to add a user to the **passwd** file. Note that the zeros in the command indicate that the user's UID and GID are both zero, which is the root user's.

Privilege Escalation
Passwd Write Permission Abuse



When we switch users to the newly made user and use the **whoami** command, we are effectively logged in as the **root** user.