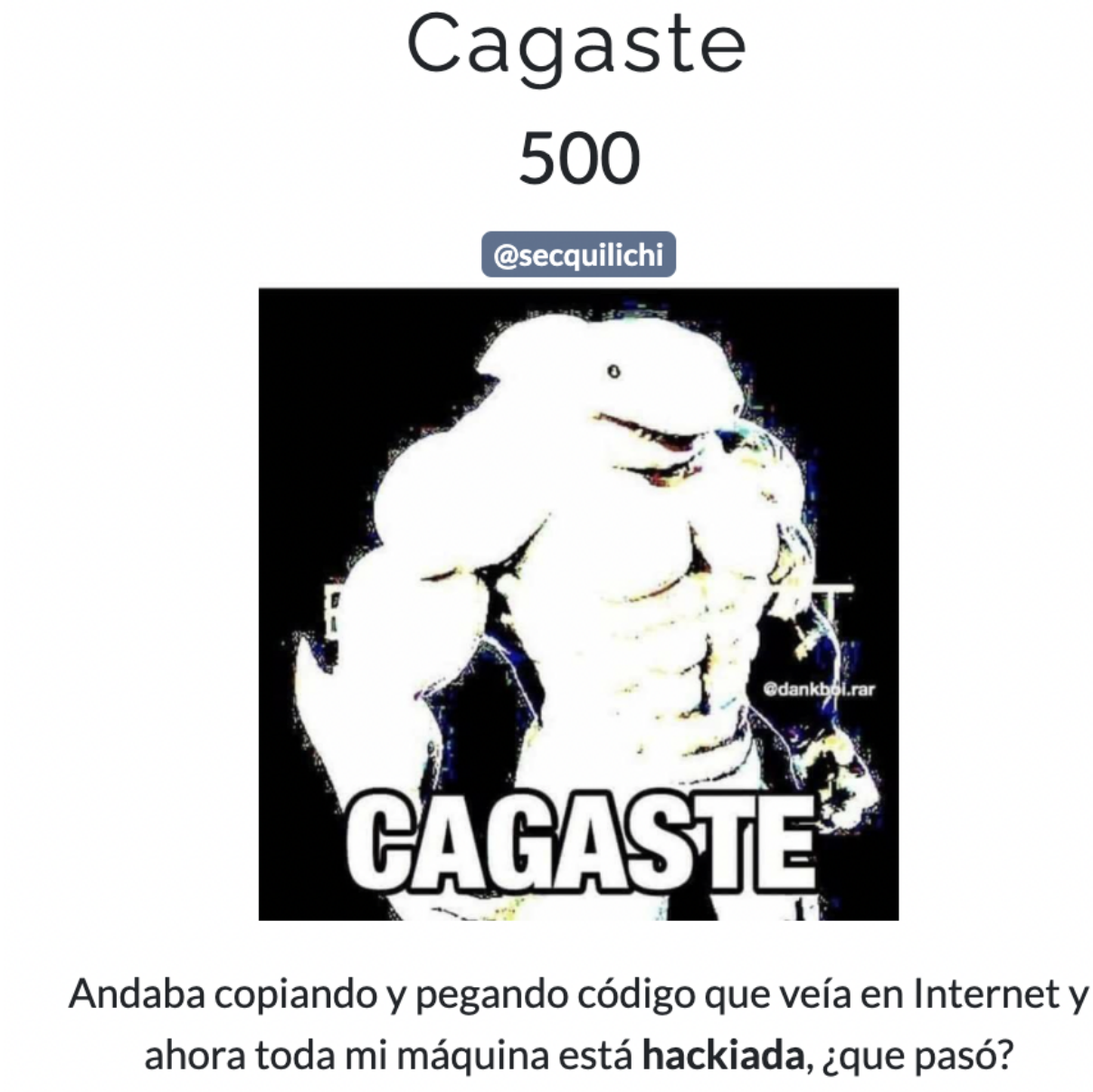
Cagaste

Categoría “Forense”

Enunciado



Solución

En el enunciado nos dicen que el usuario estaba copiando y pegando código que veía en Internet y le han atacado por culpa de dicho código. Se adjuntaba un fichero ftp_daemon.py cuyos contenidos son los siguientes

```
#!/usr/bin/env python


import atexit
import errno
import optparse
import os
import signal
import sys
import time

from pyftplib.authorizers import UnixAuthorizer
from pyftplib.filesystems import UnixFilesystem
from pyftplib.handlers import FTPHandler
from pyftplib.servers import FTPServer


# overridable options
HOST = ""
PORT = 21
PID_FILE = "/var/run/pyftpdlib.pid"
LOG_FILE = "/var/log/pyftpdlib.log"
WORKDIR = os.getcwd()
UMASK = 0


def pid_exists(pid):
    """Return True if a process with the given PID is currently running."""
    try:
        os.kill(pid, 0)
    except OSError as err:
        return err.errno == errno.EPERM
    else:
        return True


def get_pid():
    """Return the PID saved in the pid file if possible, else None."""
    try:
        with open(PID_FILE) as f:
            return int(f.read().strip())
    except IOError as err:
        if err.errno != errno.ENOENT:
            raise
```

```
def stop():
    """Keep attempting to stop the daemon for 5 seconds, first using
    SIGTERM, then using SIGKILL.
    """
    pid = get_pid()
    if not pid or not pid_exists(pid):
        sys.exit("daemon not running")
    sig = signal.SIGTERM
    i = 0
    while True:
        sys.stdout.write('.')
        sys.stdout.flush()
        try:
            os.kill(pid, sig)
        except OSError as err:
            if err.errno == errno.ESRCH:
                print("\nstopped (pid %s)" % pid)
                return
            else:
                raise
        i += 1
        if i == 25:
            sig = signal.SIGKILL
        elif i == 50:
            sys.exit("\ncould not kill daemon (pid %s)" % pid)
        time.sleep(0.1)


def status():
    """Print daemon status and exit."""
    pid = get_pid()
    if not pid or not pid_exists(pid):
        print("daemon not running")
    else:
        print("daemon running with pid %s" % pid)
    sys.exit(0)


def get_server():
    """Return a pre-configured FTP server instance."""
    handler = FTPHandler
    handler.authorizer = UnixAuthorizer()
    handler.abstracted_fs = UnixFilesystem
    server = FTPServer((HOST, PORT), handler)
    return server


def daemonize():
    """A wrapper around python-daemonize context manager."""
    def _daemonize():
        pid = os.fork()
        if pid > 0:
```

```
            # exit first parent
            sys.exit(0)

        # decouple from parent environment
        os.chdir(WORKDIR)
        os.setsid()
        os.umask(0)

        # do second fork
        pid = os.fork()
        if pid > 0:
            # exit from second parent
            sys.exit(0)

        # redirect standard file descriptors
        sys.stdout.flush()
        sys.stderr.flush()
        si = open(LOG_FILE, 'r')
        so = open(LOG_FILE, 'a+')
        se = open(LOG_FILE, 'a+', 0)
        os.dup2(si.fileno(), sys.stdin.fileno())
        os.dup2(so.fileno(), sys.stdout.fileno())
        os.dup2(se.fileno(), sys.stderr.fileno())

        # write pidfile
        pid = str(os.getpid())
        with open(PID_FILE, 'w') as f:
            f.write("%s\n" % pid)
        atexit.register(lambda: os.remove(PID_FILE))

    pid = get_pid()
    if pid and pid_exists(pid):
        sys.exit('daemon already running (pid %s)' % pid)
    # instance FTPd before daemonizing, so that in case of problems we
    # get an exception here and exit immediately
    server = get_server()
    _daemonize()
    server.serve_forever()


def main():
    global PID_FILE, LOG_FILE
    USAGE = "python [-p PIDFILE] [-l LOGFILE]\n\n" \
            "Commands:\n  - start\n  - stop\n  - status"
    parser = optparse.OptionParser(usage=USAGE)
    parser.add_option('-l', '--logfile', dest='logfile',
                      help='the log file location')
    parser.add_option('-p', '--pidfile', dest='pidfile', default=PID_FILE,
                      help='file to store/retreive daemon pid')
    options, args = parser.parse_args()

    if options.pidfile:
        PID_FILE = options.pidfile
    if options.logfile:
```

```
        LOG_FILE = options.logfile

    if not args:
        server = get_server()
        server.serve_forever()
    else:
        if len(args) != 1:
            sys.exit('too many commands')
        elif args[0] == 'start':
            daemonize()
        elif args[0] == 'stop':
            stop()
        elif args[0] == 'restart':
            try:
                stop()
            finally:
                daemonize()
        elif args[0] == 'status':
            status()
        else:
            sys.exit('invalid command')


if __name__ == '__main__':
    sys.exit(main())
```

Como podemos ver, el código lo que hace más o menos es ejecutar un servidor FTP local, pero nada más allá de eso. Cuando queremos ejecutarlo, nos tenemos que instalar la librería “pyftplib” con pip.

```
pip install pyftplib
```

Después de instalar este paquete, nos sigue dando error de paquetería al ejecutar el programa... ¿sospechoso?

La librería era el punto clave, ya que la librería de verdad se llama “pyftp**d**lib” (notese esa ‘d’).

Para obtener la flag bastaría con visitar la página web https://pypi.org/project/pyftplib/ que contiene el código de la librería maliciosa y en el famoso fichero “setup.py” encontramos la flag encodeada en base64 (para todos aquellos amantes del cntl + f masivo jejeje)

```
HackOn{sUm1n1strAm3_eSt4_bRo}
```