Dr. StrangeBug, or How I Learned to Stop Worrying and Love the Vulns

Setup

Download and install the ex07 ex_pack:
ex_unpack ex07.bin ex07

Exercise

1. Run the q1_server.py network daemon: "./q1_server.py"
   When run, this daemon will listen for connections at port 1337 (use netcat to connect: "nc 127.0.0.1 1337").
   The server is console oriented and fairly self-explanatory.
   a. Explore the interface and the code behind it.
   b. Look for an information leakage vulnerability. Document this vulnerability in a file named "q1_vuln.txt"
   c. Write python code to exploit the vulnerability in a file named "q1_infoleak.py" (your code must take an argument at the command line, and output to stdout whether the supplied PID is running or not, **without using the built-in credentials (this applies to almost all other elements in this exercise as well**))

      NOTE: **There is a corner case for processes which contain "grep" or "py" in their command line.** Please make sure you test your code against processes with names that avoid these combinations (a good example would be the init process with PID 1 as a process that is always running, and an invalid PID (anything higher than 65535) as a process that is never there).
      This applies to the rest of the exercise as well.

2. Run the q2_server.py network daemon: "./q2_server.py" (make sure to kill the q1 server first).
   This is an updated version of the previous server, which attempts to remove the vulnerability completely.
   a. Find a way to run any command, **without using the built-in credentials**.
      You may need to brush up on your shell scripting skills – use the open documentation for "sh" and "bash" (if google fails you – run "man bash").
      Document the vulnerability in q2_rce_vuln.txt, and provide python code to exploit it as a file named "q2_rce.py" (your code will take a single argument at the command line which will describe the command to be run).

   b. Given 1-8 characters for the username and 0-8 characters for the password makes brute forcing the login credentials very hard. Find a way to make brute-forcing the login credentials (username & password) more feasible.
      Document your proposed solution in a file named "q2_login_bruteforce.txt" and explain why it makes a dramatic difference in the length of time required to attempt to Brute

force the credentials. **There is no need to implement the brute-forcing code, but make sure your explanation reflects a proper understanding of how one may implement such a solution.**

c. Find a way to run any command and get its output **(this time – you may use the built-in credentials from the code)**.
Provide python code to exploit it as a file named "q2_rce_with_stdout.py" (your code will take a single argument at the command line which will describe the command to be run, and output the command's results to stdout).

d. **Using the built-in credentials**, find a way to read any local file (within the server process' permissions).
Document the vulnerability in q2_remote_file_read_vuln.txt, and provide python code to exploit it as a file named "q2_remote_file_read.py" (your code will take a single argument at the command line which will describe the absolute path (full path) to the file whose contents will be output to stdout).

<u>**Important note –**</u> this part must be done by exploiting a different vulnerable part of the code from the previous section (so please don't try to just use the previous section to "cat %s"…)

e. Find a new way to re-create your ability from the previous exercise (to tell if a process is running or not, **without using the built-in credentials**). Your solution should be indirect.
Document this vulnerability in a file named "q2_process_status_leak.txt".

f. Write python code to exploit the vulnerability in a file named "q2_process_status_leak.py" (your code must take an argument at the command line, and output to stdout whether the supplied PID is running or not).
You may want to add a few baseline tests and average your results to provide a reliable solution.

3. Run the q3_server.py network daemon: "./q3_server.py" (make sure to kill the q1/q2 servers first).
This is an updated version of the previous server, which attempts to remove the RCE vulnerability completely.

   a. Find a way to run any command and get its output **(you may use the built-in credentials)**.
   You may need to brush up on your shell scripting skills – use the open documentation for "sh" and "bash" (if google fails you – run "man bash").
   Provide python code to exploit it as a file named "q3_rce_with_stdout.py" (your code will take a single argument at the command line which will describe the command to be run, and output the command's results to stdout).