A Better Scripted Netcat Server

Brendan Sechter

2016-02-22

The last scripted netcat server I wrote about was based on a one line webserver. It was a hack that kind of worked in a pinch, but the blind loop made it a bad server.

This post covers a scripted netcat server that waits for the request header to terminate before sending a response. This makes the script well behaved.

Software Versions

```
$ date
February 18, 2016 at 07:53:00 AM JST
$ uname -vm
FreeBSD 11.0-CURRENT #0 r287598: Thu Sep 10 14:45:48 JST 2015     root@:/usr/obj/usr/src/sys/MI
```

Instructions

A well behaved scripted JSON/HTTP date server is below.

- The script executes in a temporary directory. When the script is killed, this directory is deleted.
- The temporary directory and the netcat parameters are printed when the script starts.
- The request is stored in a regular file. The response travels through a named pipe back through netcat to the client. In theory the request could be parsed or every request could be stored in a different file.
- The response is generated in a shell function so the output can easily be customized.
- The request number is printed before every request executes. Lines that come in are prepended with "<" and echoed to the terminal. Outgoing lines are prepended with ">" and echoed.
- Terminal output is printed and the request is sent only after a blank request line is received. The date is calculated only after receiving the request.

date_server.sh

```
#/bin/sh

HOST=0.0.0.0
PORT=4000

REQUEST=request
RESPONSE=response.pipe

TMP_DIR=$(mktemp -d /tmp/$(basename $0).XXXXXX) || exit 1
trap 'rm -rf "${TMP_DIR}"; exit' INT TERM QUIT EXIT
cd "${TMP_DIR}"
touch "${REQUEST}"
```

```
mkfifo -m 600 "${RESPONSE}"

echo "Path:   $(pwd)"
echo "Server: nc -N -l ${HOST} ${PORT}"

response()
{
CONTENT_TYPE=application/json
BODY=$(cat <<EOF
{
  "Date": "$(date)"
}
EOF
)
DOCUMENT=$(cat <<EOF
HTTP/1.0 200 OK
Content-Type: ${CONTENT_TYPE}
Content-Length: $((${#BODY}+1))
Connection: close

${BODY}
EOF
)
echo "${DOCUMENT}"
}

while true
do
  echo "---$((REQUEST_NUMBER=REQUEST_NUMBER+1))---"
  cat "${RESPONSE}" | nc -N -l "${HOST}" "${PORT}" |
  (
    echo -n "" >"${REQUEST}"
    while read LINE
    do
      echo "${LINE}" >>"${REQUEST}"
      LINE=$(echo "${LINE}" | tr -d '[\r\n]')
      if [ "x${LINE}" = "x" ]
      then
        cat "${REQUEST}" | sed -u 's/^/< /';
        response | tee "${RESPONSE}" | sed -u 's/^/> /';
        break
      fi
    done
  )
done
```

A single file server is like the date server. There are only a few changes.

- A file variable has been added and the content type variable has been moved.
- The script gets the full path of the variable before moving into the temporary directory.
- The body variable reads from a file instead of creating dynamic content.

file_server.sh

```
#/bin/sh
```

```
HOST=0.0.0.0
PORT=4000
FILE=index.html
CONTENT_TYPE=text/html

REQUEST=request
RESPONSE=response.pipe

FILE=$( readlink -f "${FILE}" )
TMP_DIR=$(mktemp -d /tmp/$(basename $0).XXXXXX) || exit 1
trap 'rm -rf "${TMP_DIR}"; exit' INT TERM QUIT EXIT
cd "${TMP_DIR}"
touch "${REQUEST}"
mkfifo -m 600 "${RESPONSE}"

echo "Path:   $(pwd)"
echo "Server: nc -N -l ${HOST} ${PORT}"

response()
{
BODY=$(cat "${FILE}")
DOCUMENT=$(cat <<EOF
HTTP/1.0 200 OK
Content-Type: ${CONTENT_TYPE}
Content-Length: $((${#BODY}+1))
Connection: close

${BODY}
EOF
)
echo "${DOCUMENT}"
}

while true
do
  echo "---$((REQUEST_NUMBER=REQUEST_NUMBER+1))---"
  cat "${RESPONSE}" | nc -N -l "${HOST}" "${PORT}" |
  (
    echo -n "" >"${REQUEST}"
    while read LINE
    do
      echo "${LINE}" >>"${REQUEST}"
      LINE=$(echo "${LINE}" | tr -d '[\r\n]')
      if [ "x${LINE}" = "x" ]
      then
        cat "${REQUEST}" | sed -u 's/^/< /';
        response | tee "${RESPONSE}" | sed -u 's/^/> /';
        break
      fi
    done
  )
done
```

The following file can be used for testing the file server.

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Hello World!</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

The following command can be used to test either server from the command line.

```
curl -v 127.0.0.1:4000
```

References:

- FreeBSD, Single Line Web Server With nc on FreeBSD
- UNIX, Minimal web server using netcat
- UNIX, Shell Loop Control
- UNIX, How can I concatenate string variables in Bash?
- UNIX, How to get full path of a file?