Report 1: Network System Call testing

Explicit System Call Testing

The test application would trigger all Syscalls one by one, evaluating that the audit record contains all the expected parameters, e.g the arguments, valid argument types, return values etc. The testing will be done for various success and failure modes, with cross checking for appropriate error codes in case of failure mode.

Repository

AuditTestSuite

Directory Structure

Source contains following significant files

src/sockets

- **tcp_socket.c** : Implementation of basic TCP socket which fires off a series of network syscalls. Each function is called twice, with the socket file descriptor being incorrect in one of the case, resulting in an expected error. Attempt is made to log both instances of each system call and then check whether the audit daemon logs them with the appropriate success and error message along with correct arguments.
- **udp_socket.c** : Pair of source files to launch `recvmsg(2)` and `sendmsg(2)` functions for testing UDP socket audit.
- **test** : A POSIX compliant shell script which does all the hard work. From firing off the network binary to extracting the data from resulting trail and analysing the result. Detailed functioning of the script is described later.

src

- **setup** : A script to setup the environment. i.e, start the audit daemon in case its not already running and setting up the correct flag, `flags:all` in the file `audit_control`.

Approach

Here is how I attempted to test the network syscalls.

1) Get all audit_events with network(nt) class

```
$ cat /etc/security/audit_event | grep ":nt"
32:AUE_CONNECT:connect(2):nt
33:AUE_ACCEPT:accept(2):nt
34:AUE_BIND:bind(2):nt
35:AUE_SETSOCKOPT:setsockopt(2):nt
```

```
46:AUE_SHUTDOWN:shutdown(2):nt
173:AUE_ONESIDE:one-sided session record:nt
183:AUE_SOCKET:socket(2):nt
184:AUE_SENDTO:sendto(2):nt
186:AUE_SOCKETPAIR:socketpair(2):nt
187:AUE_SEND:send(2):nt
188:AUE_SENDMSG:sendmsg(2):nt
189:AUE_RECV:recv(2):nt
190:AUE_RECVMSG:recvmsg(2):nt
191:AUE_RECVFROM:recvfrom(2):nt
216:AUE_PUTMSG:putmsg(2):nt
217:AUE_GETMSG:getmsg(2):nt
218:AUE_PUTPMSG:putpmsg(2):nt
219:AUE_GETPMSG:getpmsg(2):nt
247:AUE_SOCKACCEPT:getmsg-accept:nt
248:AUE_SOCKCONNECT:putmsg-connect:nt
249:AUE_SOCKSEND:putmsg-send:nt
250:AUE_SOCKRECEIVE:getmsg-receive:nt
265:AUE_SOCKCONFIG:configure socket:nt
288:AUE_NTP_ADJTIME:ntp_adjtime(2):ad
317:AUE_DARWIN_SOCKETPAIR:socketpair(2):nt
43054:AUE_SENDFILE:sendfile(2):nt
43140:AUE_LISTEN:listen(2):nt
43207:AUE_BINDAT:bindat(2):nt
43208:AUE_CONNECTAT:connectat(2):nt
```

Description of test.sh

2) Set the audit flag:lo,nt (login-logout, network)

```
sed -i "" '/\<flags:/s/\(.*\)/flags:lo,nt/' /etc/security/audit_control
```

3) Start the audit daemon and set a new trail for recording the syscalls

```
$ service auditd start; audit -n
```

4) Fire off the syscalls, it will create a TCP socket server. Connect to the socket with telnet and send a test message

```
./network &
telnet localhost PORT_NO | echo "Message"
```

5) Close the auditing and catch the trail which recorded the logs. Convert it into a human readable form by `praudit`. To identify the success or failure messages, it is recommended to output each token in a single line with `praudit -l LOGFILE`.

6) Loop through the logfile and search for each system call in the database. And for each one of

them, check for the presence of "return,success" and "return,failure". Presence of these texts ensure that the launch of system calls has been logged successfully in both scenarios.

7) Cleanup the test trails.

Result

Audit of all concerned test cases was successful (Yay!)

```
$ ./test.sh
Audit Directory: /var/audit .. □
Starting auditd.
Audit daemon and new trail started .. □
Launching system calls .. □
Connected via client .. □
Audit daemon stopped .. □

Trigger sent.
===============================================
Testing socket(2)..
Success mode passed: socket(2) .. □
Failure mode passed: socket(2) .. □
===============================================
Testing setsockopt(2)..
Success mode passed: setsockopt(2) .. □
Failure mode passed: setsockopt(2) .. □
===============================================
Testing bind(2)..
Success mode passed: bind(2) .. □
Failure mode passed: bind(2) .. □
===============================================
Testing listen(2)..
Success mode passed: listen(2) .. □
Failure mode passed: listen(2) .. □
===============================================
Testing accept(2)..
Success mode passed: accept(2) .. □
Failure mode passed: accept(2) .. □
===============================================
Testing sendto(2)..
Success mode passed: sendto(2) .. □
Failure mode passed: sendto(2) .. □
===============================================
Testing recvfrom(2)..
Failure mode passed: recvfrom(2) .. □
Success mode passed: recvfrom(2) .. □
===============================================
Testing connect(2)..
Success mode passed: connect(2) .. □
Failure mode passed: connect(2) .. □
===============================================
Testing sendmsg(2)..
Success mode passed: sendmsg(2) .. □
Failure mode passed: sendmsg(2) .. □
===============================================
Testing recvmsg(2)..
Failure mode passed: recvmsg(2) .. □
Success mode passed: recvmsg(2) .. □

------------------Statistics-------------------
Tests evaluated: 20
Tests passed: 20
```

Further plan and Improvements

- Separate the initial setup so that code is not repeated in the cases to be tested next. [**Done**]
- Implement filesystem call testing, i.e open(2), close(2), write(2), read(2) etc.
- Record and display Statistics of passed and failed tests. [**Done**]
- [IMP] Add the check for correct file descriptor format, argument types and logging of the other important info in the trails.

Bugs

- Need to separate the part of script which sets up the flag and audit daemon configuration. More often than not, it does not get time to initialize the daemon and hence, it is not able to record the calling of syscalls. [**Resolved**]
- Variable scoping is causing an issue. If the tests fail in the above scenario, it is not shown in the output. [**Resolved**]: Use an external file.

> **Edit**: Test for connect(2) added.!
> **Edit2**: Print statistics along with nice formatting