Snort exercise

Offensive Technologies 2021
Matteo Franzil <matteo.franzil@studenti.unitn.it>

November 14, 2021

Contents

1 Solution

1.1 Topology

This photo shows the configuration of the nodes.

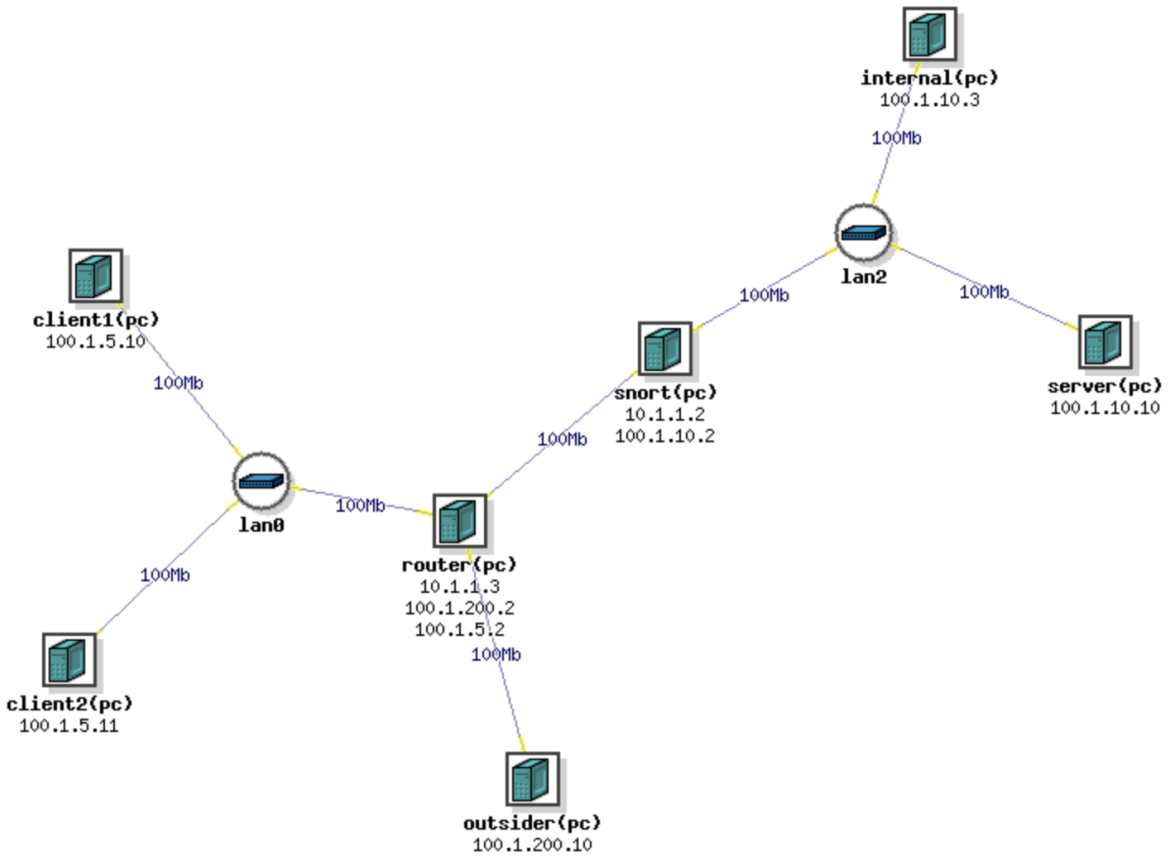

Figure 1 Network setup for the exercise.

1.2 Basic exercises

1.2.1 Start Snort Without rules

Connect to `users.deterlab.net`. SSH into the `snort` experiment node.

Start snort without any rules by entering the command `sudo snort --daq nfq -Q -v`. You should see a large number of packets being reported by Snort.

Open another terminal and SSH again into `users.deterlab.net` and the `snort` node. In this terminal run `tcpdump` to capture the data going to and from the `client1`. This should be on the interface with an IP Address in the `10.1.1.0/24` range.

You can run tcpdump using the command:

```
# tcpdump -i [interface] -s 0 -w /tmp/dump.pcap
```

Code 1 Code for using tcpdump

Allow this to run for 30 seconds and then tcpdump by pressing `^C`. Let the Snort process continue running. Now run

```
# /share/education/SecuringLegacySystems_JHU/process.pl /tmp/dump.pcap
```

Code 2 Processing the obtained data.

This will produce a set of x, y coordinates where x is time and y is number of packets in that second.

Question 1.2.1. *Respond to the following points.*

1. *What happens to the traffic to client1 when Snort is not running?*
2. *Is this a good thing?*
3. *Based on Snort's output what can you say about the application? What port does it connect to?*
4. *Please attach a graph of the traffic over time to your answers*
5. *What does the "-Q" option do in Snort?*
6. *What does the "–daq nfq" option do in Snort?*

Answer. These are the answers.

1. When snort is not running, all traffic gets dropped.
2. Not really. We can see that there are `client2` and `outsider` that are continuosly sending traffic to `server`, but it also gets dropped when snort is not running. Since Snort is woking in inline mode, all traffic is supposed to pass through the snort bridge.
3. The application accepts incoming connections on port `7777`.
4. See the graph at the end of the numbered section.
5. The -Q option tells snort to process packets in inline mode.
6. The –daq nfq option tells snort to use the `nfq` DAQ module.

□

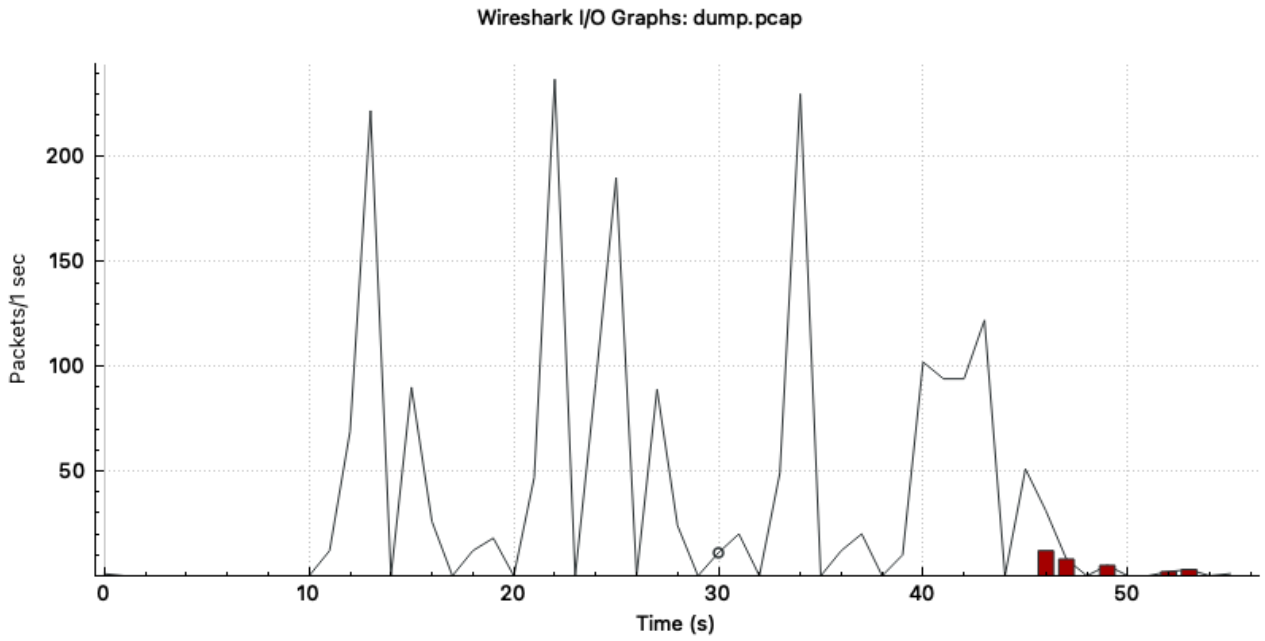

Figure 2 Traffic over time.

1.2.2 Analyze Network Traffic

Connect to the `router` node. Use `ifconfig` to determine which network interfaces connect to which network. Run `tcpdump` to capture the data going to the server. This should be on the interface with an IP Address in the `10.0.1.0/24` range.

You can run tcpdump with the same command as in Part 1. Let this run for around a minute before terminating it by pressing `^C`. Copy this data to your computer via SCP and open it using Wireshark.

Question 1.2.2. *The request that the client sends the server is broken into four parts. What are these parts and what order does they appear in? How are these parts seperated in the request? Is this is a secure way for the client to send requests to the server? Explain your answer. Can you recover one of the files sent by the server to a client? If so attach the file, a pcap the relevant packets and indicate which client this was sent to.*

Answer. □

1.2.3 Write Rules to Guard Against Simple Requests

In that terminal where snort is running stop it with `^C`. Write a configuration file for snort using the command, named `snort.config`. Add the following snort rule that prevents .xml files from being sent out.

```
reject tcp 100.1.0.0/16 ANY -> 100.1.10.10 [Port from Question 3]
    (msg: "XML Read Attempt Detected"; sid:1; content:".xml";)
```

Code 3 Code to add to the config.

Using this rule as an example write a rule that prevents classified data from being sent to the outsider computer, but does not prevent it from being sent to any other computers. Now you must make a directory for snort alerts, called `alerts`. Start snort using your new rule with the command:

```
# snort --daq nfq -Q -c snort.config -l alerts
```

Code 4 Code to start snort.

Log onto `client1`, `client2` and `outsider` to see if these rules worked. If you aren't sure go to the folder `/home/test` and delete the existing .txt and .xml files then see which ones are recreated. You can also run the program FileClient manually to find out. Questions:

Question 1.2.3. *Respond to the following points.*

- *What rule did you use to secure the "classified" file?*
- *Capture and compare the network traffic for the serverwhen filtering these results using your configuration file and when no file is used. Attach the graph showing packet rate over time for both of these cases to your submission.*
- *Can you think of any others files or extensions that should be filtered against?*

1.3 Intermediate tasks

1.3.1 DOS defense

Make sure you have your filtering rule engaged on Snort. Using the `flooder` tool on client1 create a packet flood attack against the server from. Set a high rate of 100,000 requests and duration of 100 seconds. Just typing flooder on command line should give you a list of options you can use.

Create a new Snort configuration that also includes a rate filter rule. These follow the following format:

```
rate_filter gen_id 1, sig_id [sid of event],
track [by_src, by_dst or by_rule], count [number of events],
seconds [number of seconds], new_action drop,
timeout [number of seconds]
```

Code 5 Format for rate filters.

You may need to write a new rule for the rate filter to apply to.

Create an event filter for your rate filter. These follow the format:

```
event_filter gen_id 1, sig_id [sid_number],
track [by_src, by_dst or by_rule], count [number of events],
seconds [number of seconds],
type [limit, threshold or both]
```

Code 6 Format for event filters.

Question 1.3.1. *Respond to the following points.*

1. *Collect the traffic at the server and the router when there is and is not an attack with rules in place that only guard against the attacks metioned in the basic exercises. Create a graph of this traffic over time.*
2. *Repeat the previous step but now with the rate filtering rules enabled.*
3. *For a DOS attack should rate filtering rules be paired with event filtering rules?*
4. *Try changing the new action in the rate filter to "reject" instead of "drop". What does this do to the traffic and why do you think this is? Is this a good or a bad thing?*
5. *Check which interface Snort connects to the router to using ifconfig. Once you have this information try the above test against while specifying that interface instead of using the default value. This argument will look something like –daq-var device=eth1. Did this change your results. If so attach a graph and explain the change occured.*
6. *Are there any changes you can make outside of Snort to help guard against this attack? If so what are they?*
7. *Try running the FileClient program from client2 while this attack is underway. What happens when you have the various rulesets configured?*

1.3.2 Secure Traffic on the Network

Make sure that Snort is running with all your filtering rules in place. Now, review the pcap data you gathered while analyzing network traffic on the router.

Review the network topology on DETER. Notice that the internal computer is able to bypass Snort entirely!

Verify this by logging onto the internal computer and checking the folder `/home/test`. You should see that it is able to freely download files that your rules do not permit.

Replace the direct route to the server using the following command:

```
# route add -host server gw snort
```

Code 7 Code for replacing the direct route to the server.

This should cause traffic to the server to be routed through Snort. Perform a trace route to verify this with the command `traceroute server`.

Question 1.3.2. *Respond to the following points.*

1. *Based on the traffic you analyzed what changes could be made to the network to enhance the security of communications coming from client1, client2 and outsider?*
2. *What software packages would this require and where should these be installed? Would this cause problems for Snort?*
3. *How should the server be configured to prevent internal attacks?*
4. *Would this require you to change your Snort configuration in any way?*

1.4 Advanced Tasks

1.4.1 Code Execution Vulnerability

A vulnerability was found with the FileServer that causes it to execute the contents of the filename field if certain conditions are met.

To do this point, you must download the file `/home/test/FileServer.jar` from the server. Analyze the file using a Java decompiler such as JD-GUI to find the vulnerability.

Question 1.4.1. *Respond to the following points.*

1. *What are the conditions required for this attack to take place?*
2. *Create a Snort rule to defend against this attack. You may want to be use pcre instead of content for this rule.*
3. *What effect does this rule have on legitimate traffic?*

1.4.2 Defend Against ASCII Encoding

An additional "feature" was recently brought to the attention of the security team: the application has a limited ability to before ASCII decoding. This decoding is triggered whenever the server encounters a % followed by a number between 0 and 255 at which point the sequence is converted to a single ASCII character.

Now go to the following directory:

`/usr/local/snort-2.9.2.2/src/dynamic-examples/dynamic-preprocessor`

This contains an example Snort preprocessor. Snort preprocessors allow for advanced data manipulation to normalize data and perform other functions that normal rules cannot. All Snort preprocessors are coded in C.

Open up `spp_example.c` and take a look at the structure. The main function of interest to you in this case is `ExampleProcess` which is used to process packet data before other rules have a chance to access it. Copy the include folder using the following command:

```
# cp /usr/local/snort-2.9.2.2/src/dynamic-preprocessors/include/ ..
```

Code 8 Code for including the folder.

You should now be able to compile and install this preprocessor using the command:

```
# make && make install
```

Code 9 Make command.

Once this has been installed you can use this preprocessor in Snort. To do this edit your configuration file by adding the following lines at the top of the file which tell Snort to auto-generate the decoder rules so you don't have to, the location of the preprocessor library and the preprocessor you wish to use along with all of its arguments (in this case just the port to use):

```
config autogenerate_preprocessor_decoder_rules
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
preprocessor dynamic_example: port [Port from Question 3]
```

Code 10 Code for adding the preprocessor to the configuration file.

Once you have made these changes try to run Snort using your configuration file. At this point it should behave exactly as before, but it now has a preprocessor loaded which you can edit to meet your needs as an ASCII decoder.

Question 1.4.2. *Respond to the following points.*

1. *Were you able to bypass your existing rules because of this feature? If so what input strings did you use?*
2. *Can you think of a content rule to effectively defend against an attack that uses this feature? Would this affect legitimate traffic?*
3. *Snort includes support for user written preprocessors that can render data for Snort's other rules. How would the use of a preprocessor help with this task?*
4. *Write a preprocessor to help with this task. Please attach all of the functions you used and the snort.config file that called the preprocessor.*