Summary

Implement a small program that reads `/proc/net/tcp` every 10 seconds and outputs any new connections.

Requirements

This challenge covers 5 engineering levels at Teleport. Level 6+ is reserved for internal promotions and other engineering roles.

We've decided we want to put together a small program to report new TCP connections on a linux host. Write a program that meets the requirements of the level you are applying for, and submit the results. Also please submit answers to the questions outlined in each level.

/proc/net/tcp format

The format is described here: https://www.kernel.org/doc/Documentation/networking/proc_net_tcp.txt

```
$ cat /proc/net/tcp
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt   uid  timeout i
   0: 00000000:1F99 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 3
   1: 00000000:DFF9 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 2
   2: 00000000:BDDB 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 2
   3: 00000000:0801 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 1
   4: 0100007F:8287 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 2
   5: 00000000:98CB 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 1
   6: 00000000:006F 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 1
   7: 00000000:B315 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 2
   8: 3500007F:0035 00000000:0000 0A 00000000:00000000 00:00000000 00000000   102        0 1
   9: 00000000:0016 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 2
  10: E10FA20A:8CF8 0F5BBD5B:0050 06 00000000:00000000 03:00001212 00000000     0        0 0
  11: E10FA20A:CCA0 6E0DD9AC:0050 06 00000000:00000000 03:00001211 00000000     0        0 0
  12: E10FA20A:DC1A FEA9FEA9:0050 01 00000000:00000000 00:00000000 00000000     0        0 2
  13: E10FA20A:DC1E FEA9FEA9:0050 01 00000000:00000000 00:00000000 00000000     0        0 2
  14: E10FA20A:D8B4 9858BD5B:0050 06 00000000:00000000 03:00001219 00000000     0        0 0
  15: E10FA20A:AD6C 145CBD5B:01BB 01 00000000:00000000 00:00000000 00000000     0        0 2
  16: E10FA20A:DC1C FEA9FEA9:0050 01 00000000:00000000 00:00000000 00000000     0        0 2
  17: E10FA20A:E4F4 2A5BBD5B:01BB 01 00000000:00000000 00:00000000 00000000     0        0 2
  18: E10FA20A:0016 3BD7E1CC:D94B 01 00000000:00000000 02:000AF7F9 00000000     0        0 2
  19: E10FA20A:E6A6 B358BD5B:01BB 01 00000000:00000000 00:00000000 00000000     0        0 2
  20: E10FA20A:D8B2 9858BD5B:0050 06 00000000:00000000 03:00001219 00000000     0        0 0
```

The second and third columns contain IP addresses and port pairs in hex format. Be careful since the IP address is printed in little endian, and the port is printed in big endian (on x86 architectures).

0100007F:0050 translates to 127.0.0.1:80 E10FA20A:01BB translates to 10.162.15.225:443

Level 1:

Description

Write a program that reads `/proc/net/tcp` every 10 seconds, and reports any new connections.

Sample Output:

```
2021-04-28 15:28:05: New connection: 192.0.2.56:5973 -> 10.0.0.5:80
2021-04-28 15:28:05: New connection: 203.0.113.105:31313 -> 10.0.0.5:80
2021-04-28 15:28:15: New connection: 203.0.113.94:9208 -> 10.0.0.5:80
2021-04-28 15:28:15: New connection: 198.51.100.245:14201 -> 10.0.0.5:80
```

Include a readme with the program that explains any dependencies and how to build and execute the program. The interview panel will build and test the program.

Questions

1. How would you prove the code is correct?
2. How would you make this solution better?
3. Is it possible for this program to miss a connection?
4. If you weren't following these requirements, how would you solve the problem of logging every new connection?

Level 2:

Implement all of the level 1 requirements plus: 1. Add a Makefile or your preferred build scripting to build and test the solution. 2. Add some tests, test for corner cases or unexpected behaviour. 3. Add the ability to detect a port scan, where a single source IP connects to more than 3 host ports in the previous minute.

Sample Output:

```
2021-04-28 15:28:05: Port scan detected: 192.0.2.56 -> 10.0.0.5 on ports 80,81,82,83
```

Questions

1. Why did you choose `x` to write the build automation?
2. Is there anything else you would test if you had more time?
3. What is the most important tool, script, or technique you have for solving problems in production? Explain why this tool/script/technique is the most important.

Level 3:

Implement all of the level 2 requirements plus: 1. Add a prometheus endpoint to report metrics on the following: 1. Counter - number of new connections

`Tip: There are client libraries available in many languages for providing prometheus metrics`

2. Build the project into a docker container, provide instructions on how to execute as a container while reporting connections on the host.
3. When a port scan is detected, configure the host firewall to block connections by source-ip.

Questions

1. If you had to deploy this program to hundreds of servers, what would be your preferred method? Why?
2. What is the hardest technical problem or outage you've had to solve in your career? Explain what made it so difficult?

Level 4:

Instead of polling `/proc/net/tcp` or the host for a list of connections it's tracking, we want to use a better model and track connection attempts in real time. Use one of the following methods to track attempted connections: 1. Use a pcap library to track TCP SYN Packets 1. Make sure the capture has a filter set to limit the number of packets passed to userspace to only interesting packets required for connection tracking. 2. Load a BPF program into the linux express data path (XDP) and attach to an interface to monitor for new connections and report to userspace. 3. Load a BPF program as a Linux Security Module (LSM) and report on new connections to userspace.

Note: Reading `/proc/net/tcp` is no longer required, and should not be implemented.

Level 5:

Instead of configuring the host based firewall to block the source of a port scan, write a BPF program to do so as either a LSM or XDP implementation.

Note: Writing to the host firewall is no longer required and should be replaced by the above method.

Scoring

The submission will be provided to a panel of team members, who will review the submission, run the code, try and break it, and review the written answers. Each panel member will then privately submit a vote of +1 or -2 within our tracking system along with comments on the submission. Once the results have

been collected, the hiring manager will schedule a call with the panel to go over the results, how the interview went, and come to agreement on the results.

In case of a positive result, we will connect you to our HR team who will make an offer and coordinate the rest of the hiring process.

In case of a negative result, the hiring manager will contact you via email and share a list of key observations from the panel that affected the result.

What we look for

We do our best to eliminate as many biases as possible in our interview process, and look only at the technical merits of the submission based on our experience. The panel members don't review resumes or backgrounds, and only see the submission itself and any communications over slack. Our internal voting process requires comments, so that we can try and identify individual biases that may occur.

In general we look for: - Secure solutions where appropriate, everything we do impacts the security of our customers. (Shortcuts are allowed for this project, but with commentary on how to properly implement security). - How the candidate communicates, this includes setting expectations, README's, in code documentation, etc. - How the candidate explains technical details. - The technical merits of the solution, is it cleanly written, easy to understand, does it handle corner cases, etc. - Are there any glaring problems or limitations outside of expectations. - A demonstration of intellectual curiosity - how willing the candidate is to do research into the latest and greatest, dig into the little things to find the big problems, and adapt to the ever changing tech landscape.

Interview channel

For the duration of the interview we will invite you to a slack channel with our team. Feel free to use the channel to coordinate with us, ask any questions, etc. Also, feel free to take the opportunity to ask the team questions about the company and team, it's just as important that the candidate ensure our culture and team are a good fit for them.

Use the interview channel to submit the answers to the questions and link to the code when ready. We'll use the channel to follow up on any questions, and report and issues running the submission.

Tips

- Use any programming language or tools you would normally use to complete this task in your work day.
- Include clear instructions on any dependencies so we can build and test the program ourselves.

- Add comments to help us understand the code.
- It's OK and encouraged to take shortcuts and keep the solution brief, make sure to include comments to help us understand where these shortcuts were taken.
- It should take 6-10 hours to complete the challenge, be mindful of your own time and try and avoid scope creep.
- We do not provide higher scores for submissions that are submitted quickly.
- From when the challenge is started you have a maximum of 1 week to complete the challenge.
- Keep answers to the questions brief, 4 to 8 sentences maximum.
- We claim no ownership over the submission itself, you may do what you wish with it after the interview.
- Feel free to drop the code into a github repository, and invite us as contributors if you prefer to keep the submission private.