INSTASAFE ASSIGNMENT

1: Find the bug and share the updated code with your inputs.

```
/*Start Assignment 2.1*/

         int x;
         if (x > 7) {
         x = 5;
         }
         System,.out.println(x);

/*End Assignment 2.1*/

/*Start Assignment 2.2*/
main() {
  int j;
  j = foo(i);
  ineedj();
}
/*End Assignment 2.2*/

/*Start Assignment 2.3*/
Assignment 2:
for (i=0; i<numrows; i++)
  for (j=0; j<numcols; j++);
    pixels++;

/*End Assignment 2.3*/
```

2: Write a brief notes (1 page each) on the following topics

- a: Software Defined Perimeter
- b: Synopsis of Zero Trust Network Access

3: Implement syslog on an AWS instance (Free tier)

4: Create passwordless login between 2 AWS EC2 instances

1st solution

```
/*Start Assignment 2.1*/

         int x;
         if (x > 7) {
         x = 5;
         }
         System.out.println(x);


/*Start Assignment 2.2*/
main() {
  int j;
  j = foo(i);
  ineedj();
}
/*End Assignment 2.2*/
```

2.2 This illustrates some fun with global/local variables. In function foo, j is local and i is global. Since i is being used as a loop variable, this is almost certainly wrong. Making j local here may or may not be logically correct, but it is certainly stylistically incorrect since the semantic meaning of j is being used in two distinct ways (once as a global, once as a local, which by definition must be inconsistent).

```
/*Start Assignment 2.3*/
Assignment 2:
for (i=0; i<numrows; i++);
  for (j=0; j<numcols; j++);
    pixels++;

/*End Assignment 2.3*/
```

2nd solution

a: Software Defined Perimeter

A security mechanism known as the software-defined perimeter (SDP) assigns users access to internal apps based on their identities, with context-sensitive trust. SDP, which is provided by the cloud, is everywhere, as opposed to traditional security, which is centralized in the data Centre. It plays a crucial role in protecting enterprises that priorities the cloud and mobile platforms by using business policy to determine user authentication to resources.

Defense Information Systems Agency (DISA) first proposed the idea of SDPs in 2007. SDPs are based on a need-to-know model with trust that is continuously monitored and adjusted in accordance with a number of criteria. They reduce the attack surface from network-based attacks by hiding application infrastructure from the internet (DDoS, ransomware, malware, server scanning, etc.).

In its early phases, the Cloud Security Alliance (CSA) became interested in the idea and started working on the SDP framework. When SDP was still a relatively new idea in 2011, Google became an early adopter by creating its own SDP solution, Google BeyondCorp. With the advent of work-from-anywhere policies, enterprises are now upgrading their endpoint, cloud, and application security.

SDP follows a different methodology than conventional network-based security. SDP focuses on securing the user, the application, and the connectivity in between, as opposed to securing the network. SDP technologies differ from one another in four key ways:

Trust is never assumed: Users can place too much reliance in conventional network security. Trust must be earned in an SDP. Specifically, SDPs only allow users who have been authenticated and specifically given permission to use that app to access applications. Furthermore, only the application—not the network—is accessible to approved users.

No connections coming in: SDPs don't get any inbound connections, in contrast to a virtual private network (VPN), which listens for connections from outside the network. SDPs maintain network and application architecture as invisible or cloaked to the internet by responding with outbound-only connections.

b: Synopsis of Zero Trust Network Access

Based on explicitly defined access control criteria, Zero Trust Network Access (ZTNA), an IT security solution, offers safe remote access to an organization's applications, data, and services. In contrast to virtual private networks (VPNs), which allow access to the entire network, ZTNA only allows access to particular services or apps. ZTNA solutions can aid in closing gaps in other secure remote access technologies and approaches as more users access resources remotely from their homes or other locations.

When ZTNA is in operation, access to particular programmes or resources is only given following user authentication through the ZTNA service. A secure, encrypted tunnel that adds an additional degree of security protection by hiding applications and services from IP addresses that could otherwise be visible is used by the ZTNA to grant the user access to the specific application after they have successfully authenticated.

By using the same concept of the "black cloud" to keep users from having visibility into any other apps and services they do not have authorization to access, ZTNAs behave very similarly to software defined perimeters (SDPs). Due to the fact that even if an attacker gained access, they would not be able to scan, this also provides protection against lateral attacks.

ZTNA implementation can be done in one of two ways: endpoint- or service-initiated.

In an endpoint-initiated zero trust network architecture, as the name suggests, the user initiates access to an application from a device connected to an endpoint, much like an SDP. The ZTNA controller, which connects to the required service and provides authentication, is contacted by an agent placed on the device.

In contrast, a broker between the application and the user initiates the connection in a service-initiated ZTNA. The business apps, which can be found on-premises or at cloud providers, must have a lightweight ZTNA connector in front of them in order to accomplish this. Traffic will pass through the ZTNA service provider, isolating apps, after the outbound connection from the requested application authenticates the user or another application

4th solution

Steps to create password less ssh between two AWS EC2 instances

Method 1

pre-reqs:

1.access to yours EC2 machine and using the primary key or credentials
With root permissions.

2.already setup RSA keys on your local machine. private key and public
Key and public key are available at “~/.ssh/id rsa” and
“~/.ssh/id_rsa.pub”, respectively.

steps:

1. login to Ec2machine as a root user.
2. Create a new user

```
Useradd -m <yourname>
Sudo su <yourname>
Cd
Mkdir -p ~/.ssh
Touch ~/.ssh/authorized_keys
```

Append contents of files~/.ssh/id_rsa.pub on your local machine to ~/.ssh/authorized_keys on Ec2machine.

```
Chmod -R 700 ~/.ssh
Chmod 600 ~/.ssh/*
```

3.check whether ssh-ing is permitted by the machine. It should.
In/etc/ssh/sshd_config,line containing “passwordAuthentication yes”
Is uncommented.restart sshd service if you make any change in this
File:

```
Service sshd restart # on Centos
Service sshd restart # on ubantu
```

4.your passwordless login should work now. Try following on your
Local machine:

```
Ssh -A <yourname>@ec2-xx-xx-xxx-xxx.ap-southeast-
   compute.amazonaws.com
```

5.making yourself a super usetr.open /etc/sudoers. Make sure following
Two lines are uncommented:

```
## Allows people in group wheel to run all commands
%wheel ALL=(ALL)     ALL

##Same thing without a password
%wheel ALL=(ALL)     NOPASSWD: ALL
```

Add yourself to wheel group.

```
 Usermod -aG wheel <yourname>
```

Method no 2

Pre Requisites

2 Linux Servers

Redhat Linux Server - A

IP: 13.233.128.180

User: ec2-user

Password: ****

Redhat Linux Server - B

IP: 13.127.64.6

User: ec2-user

Password: ****

Step 1

Generate the ssh key using below command in Redhat Linux Server - A

ssh-keygey

~/.ssh/

id_rsa

id_rsa.pub

Step 2

Copy the public key from Redhat Linux Server - A to Redhat Linux Server - B as follows.

```
ssh-copy-id ec-user@Redhat Linux Server - B HostName/IP Address
```

Step 3

Test the configurations as follows.

```
ssh ec2-user@Redhat Linux Server - B HostName/IP Address
```

It should not ask the password and It will successfully connected to Redhat Linux Server - B from Redhat Linux Server – A

3rd solution

Implement syslog on an AWS instance (Free tier)

Installing syslog-ng from EPEL

The following command enables the EPEL repository on Amazon Linux 2:

```
amazom-linux-extras install epel
```

From here, the process is pretty much the same as on any other RHEL 7 / CentOS 7 system. You install syslog-ng using yum, and then use systemctl to enable and start it. Finally, remove rsyslog (or at least stop and disable it) to avoid confusion, like duplicated log messages.

```
yum install syslog-ng
systemctl enable syslog-ng
```

```
systemctl start syslog-ng
yum erase rsyslog
```

Installing unofficial syslog-ng packages

Installing the unofficial syslog-ng packages is almost the same process as installing them from EPEL. The reason is that some of the dependencies are actually coming from the EPEL repository. There is just one additional step: enabling the unofficial syslog-ng repository. The current latest syslog-ng version is 3.29, so – once you enable EPEL – you can download the repo file with the following commands:

```
cd /etc/yum.repos.d/
wget https://copr.fedorainfracloud.org/coprs/czanik/syslog-ng329/repo/epel-7/czanik-syslog-ng329-epel-7.repo
```

You can check the latest available version by looking at https://copr.fedorainfracloud.org/coprs/czanik/ and read more about the various repositories at https://www.syslog-ng.com/community/b/blog/posts/overview-of-syslog-ng-rpm-repositories Note that not all repositories have Aarch64 support enabled. Let me know either on Twitter or GitHub issues if you need Aarch64, but it is not enabled for the given repository.

Once the unofficial repository is enabled, installing syslog-ng is the same process as with the basic EPEL package.

Testing

After installation, you should do at least some minimal testing. The method below works both with EPEL and unofficial packages, as it only tests local log sending using the default configuration.

```
# systemctl status syslog-ng
● syslog-ng.service - System Logger Daemon
   Loaded: loaded (/usr/lib/systemd/system/syslog-ng.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2020-11-02 13:44:28 UTC; 5s ago
     Docs: man:syslog-ng(8)
 Main PID: 9245 (syslog-ng)
   CGroup: /system.slice/syslog-ng.service
           └─9245 /usr/sbin/syslog-ng -F -p /var/run/syslogd.pid
```

Finally, send a test message to it and check if the message arrived successfully in /var/log/messages:

```
# logger this is a test
# tail -1 /var/log/messages
```

```
Nov  2 13:44:56 ip-1-2-3-4.us-east-2.compute.internal ec2-user[9250]: this is a
test
```

Once syslog-ng is up and running in your test environment, you should prepare for the next steps:

- The default syslog-ng configuration collects log messages locally, while the main strengths of syslog-ng are central log collection and its many destination drivers. Centralize your logs using syslog-ng, and after careful processing and filtering, forward them for further analysis.
- The steps above detailed how to install syslog-ng manually, while in a production environment, syslog-ng should be installed and configured using your favorite configuration management system.