Minimizing Host OS Footprint (Reduce Attack Surface) on Ubuntu on AWS

A step-by-step guide to minimize the host OS footprint and reduce the attack surface on your Ubuntu server running on AWS:

Update hosts regularly
Remove unwanted utilities
Limit access with firewall rules
Log all system activities

e.g docker makes a system call and if it is as a root then it can do anything with system.

Create a pod to use host namespace only if necessary

spec:

```
  # container will use host IPC namespace (Default is false)
  hostIPC: true
  # containers will use host network namespace (Default is false)
  hostNetwork: true
  # containers will use host pid namespace (Default is false)
  hostPID: true
  containers:
  - name: nginx
     image: nginx:latest
```

Don’t run containers in privileged mode (*privileded = false, non-root user*)

```
spec:
  securityContext:
    runAsUser: RunAsAny
    runAsGroup: RunAsAny
    fsGroup: RunAsAny
  # container will use host IPC namespace (Default is false)
  hostIPC: true
  # containers will use host network namespace (Default is false)
  hostNetwork: true
  # containers will use host pid namespace (Default is false)
  hostPID: true
  containers:
    - image: nginx:latest
      name: web
      resources: {}
      securityContext:
```

```
    # container will run as root (Default is false)
    privileged: true
```

Limit node access to users
/etc/passwd
user don’t need access to nodes so delete them or limit only to admin
maintain sudoers in
cat /etc/sudoers

Remove unnecessary binaries and services
systemctl list-units --type service
systemctl stop
systemctl disable
or
apt remove <name> -y

Control access using SSH, disable root and Password-based logins
mostly you are using jump host to access master and worker nodes.
So use proper ssh keys and disable password based ssh.
cat /home/mark/.ssh/authorized_keys

#harden ssh
PermitRootLogin no
PasswordAuthentication no

#disable services
systemctl restart sshd

Adding correct firewall rules (previous lecture)
Preventing containers from loading unwanted kernel modules
#list all kernel modules

blocklist module sctp, dccp
vi /etc/modprobe.d/blacklist.conf
blacklist sctp
blacklist dccp

#reboot
shutdown –r now

identify and address open ports
netstat -tulnp
netstat -tulnp | grep -i 9090 | grep -w -i listen

#check for service to port maping

cat /etc/services | grep -i httpd

#disable services
systemctl stop httpd

Restrict Allowed Host Path using PodSecurityPolicies
https://kubernetes.io/docs/reference/access-authn-authz/psp-to-pod-security-standards/

1. Package Management:
•**Update packages:** Run sudo apt update && sudo apt upgrade -y to ensure your system has the latest security updates and bug fixes.
•**Remove unnecessary packages:** Identify and remove any unused or unnecessary packages using dpkg -l or tools like apt-get autoremove or apt-get remove --auto-remove. Be cautious while removing packages to avoid breaking dependencies.
2. User Accounts and Permissions:
•**Disable unused accounts:** Disable or delete any user accounts that are not actively used. Use userdel command for removing accounts and passwd -l username to disable them.
•**Limit root access:** Avoid using the root account for everyday tasks. Create a dedicated user account with limited privileges for administrative tasks using sudo.
3. Services and Processes:
•**Identify and disable unnecessary services:** Use systemctl list-unit-files to list all systemd units (services) and identify those not required. Utilize systemctl disable <service-name> to disable them.
•**Monitor running processes:** Utilize tools like ps aux or htop to identify and stop any unnecessary processes.
4. Network Security:
•**Configure Firewalls:** Utilize tools like UFW (Uncomplicated Firewall) to configure firewall rules and restrict incoming and outgoing traffic to only authorized ports and services.
•**Secure SSH:** Consider disabling password-based authentication in SSH and rely on key-based authentication for improved security.
5. Additional Hardening Measures:
•**Disable unused kernel modules:** Use lsmod to list loaded modules and rmmod <module_name> to remove unnecessary ones.
•**Consider using a minimal base image:** When launching your server on AWS, choose a minimal Ubuntu image without pre-installed packages to further reduce the attack surface.

•**Regularly audit and review your security configuration:** Utilize tools like sysctl and auditd to monitor and audit system configurations and identify potential security risks.

Remember: These are general recommendations, and the specific steps may vary based on your specific environment and needs.

•Consult the official Ubuntu documentation for detailed instructions and best practices: https://help.ubuntu.com/

•Refer to the AWS security documentation for guidance on hardening your instances on AWS: https://docs.aws.amazon.com/whitepapers/latest/aws-security-best-practices/welcome.html

By implementing these steps, you can significantly reduce the attack surface of your Ubuntu server running on AWS, making it more secure and resilient against potential threats.