|  |
| --- |
| #!/bin/sh |
|  |  |
|  | logit "\n" |
|  | info "5 - Container Runtime" |
|  |  |
|  | # If containers is empty, there are no running containers |
|  | if [ -z "$containers" ]; then |
|  | info " \* No containers running, skipping Section 5" |
|  | else |
|  | # Make the loop separator be a new-line in POSIX compliant fashion |
|  | set -f; IFS=$' |
|  | ' |
|  | # 5.1 |
|  | check\_5\_1="5.1 - Verify AppArmor Profile, if applicable" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | policy=$(docker inspect --format 'AppArmorProfile={{ .AppArmorProfile }}' "$c") |
|  |  |
|  | if [ "$policy" = "AppArmorProfile=" -o "$policy" = "AppArmorProfile=[]" -o "$policy" = "AppArmorProfile=<no value>" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_1" |
|  | warn " \* No AppArmorProfile Found: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* No AppArmorProfile Found: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none without AppArmor |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_1" |
|  | fi |
|  |  |
|  | # 5.2 |
|  | check\_5\_2="5.2 - Verify SELinux security options, if applicable" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | policy=$(docker inspect --format 'SecurityOpt={{ .HostConfig.SecurityOpt }}' "$c") |
|  |  |
|  | if [ "$policy" = "SecurityOpt=" -o "$policy" = "SecurityOpt=[]" -o "$policy" = "SecurityOpt=<no value>" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_2" |
|  | warn " \* No SecurityOptions Found: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* No SecurityOptions Found: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none without SELinux |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_2" |
|  | fi |
|  |  |
|  | # 5.3 |
|  | check\_5\_3="5.3 - Verify that containers are running only a single main process" |
|  |  |
|  | fail=0 |
|  | printcheck=0 |
|  | for c in $containers; do |
|  | processes=$(docker exec "$c" ps -el 2>/dev/null | tail -n +2 | grep -c -v "ps -el") |
|  | if [ "$processes" -gt 1 ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_3" |
|  | warn " \* Too many proccesses running: $c" |
|  | fail=1 |
|  | printcheck=1 |
|  | else |
|  | warn " \* Too many proccesses running: $c" |
|  | fi |
|  | fi |
|  |  |
|  | exec\_check=$(docker exec "$c" ps -el 2>/dev/null) |
|  | if [ $? -eq 255 ]; then |
|  | if [ $printcheck -eq 0 ]; then |
|  | warn "$check\_5\_3" |
|  | printcheck=1 |
|  | fi |
|  | warn " \* Docker exec fails: $c" |
|  | fail=1 |
|  | fi |
|  |  |
|  | done |
|  | # We went through all the containers and found none with toom any processes |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_3" |
|  | fi |
|  |  |
|  | # 5.4 |
|  | check\_5\_4="5.4 - Restrict Linux Kernel Capabilities within containers" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | caps=$(docker inspect --format 'CapAdd={{ .HostConfig.CapAdd}}' "$c") |
|  |  |
|  | if [ "$caps" != 'CapAdd=' -a "$caps" != 'CapAdd=[]' -a "$caps" != 'CapAdd=<no value>' -a "$caps" != 'CapAdd=<nil>' ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_4" |
|  | warn " \* Capabilities added: $caps to $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Capabilities added: $caps to $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none with extra capabilities |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_4" |
|  | fi |
|  |  |
|  | # 5.5 |
|  | check\_5\_5="5.5 - Do not use privileged containers" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | privileged=$(docker inspect --format '{{ .HostConfig.Privileged }}' "$c") |
|  |  |
|  | if [ "$privileged" = "true" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_5" |
|  | warn " \* Container running in Privileged mode: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Container running in Privileged mode: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found no privileged containers |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_5" |
|  | fi |
|  |  |
|  | # 5.6 |
|  | check\_5\_6="5.6 - Do not mount sensitive host system directories on containers" |
|  |  |
|  | # List of sensitive directories to test for. Script uses new-lines as a separator. |
|  | # Note the lack of identation. It needs it for the substring comparison. |
|  | sensitive\_dirs='/boot |
|  | /dev |
|  | /etc |
|  | /lib |
|  | /proc |
|  | /sys |
|  | /usr' |
|  | fail=0 |
|  | for c in $containers; do |
|  | docker inspect --format '{{ .VolumesRW }}' "$c" 2>/dev/null 1>&2 |
|  |  |
|  | if [ $? -eq 0 ]; then |
|  | volumes=$(docker inspect --format '{{ .VolumesRW }}' "$c") |
|  | else |
|  | volumes=$(docker inspect --format '{{ .Mounts }}' "$c") |
|  | fi |
|  | # Go over each directory in sensitive dir and see if they exist in the volumes |
|  | for v in $sensitive\_dirs; do |
|  | sensitive=0 |
|  | contains "$volumes" "$v:" && sensitive=1 |
|  | if [ $sensitive -eq 1 ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_6" |
|  | warn " \* Sensitive directory $v mounted in: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Sensitive directory $v mounted in: $c" |
|  | fi |
|  | fi |
|  | done |
|  | done |
|  | # We went through all the containers and found none with sensitive mounts |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_6" |
|  | fi |
|  |  |
|  | # 5.7 |
|  | check\_5\_7="5.7 - Do not run ssh within containers" |
|  |  |
|  | fail=0 |
|  | printcheck=0 |
|  | for c in $containers; do |
|  |  |
|  | processes=$(docker exec "$c" ps -el 2>/dev/null | grep -c sshd | awk '{print $1}') |
|  | if [ "$processes" -ge 1 ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_7" |
|  | warn " \* Container running sshd: $c" |
|  | fail=1 |
|  | printcheck=1 |
|  | else |
|  | warn " \* Container running sshd: $c" |
|  | fi |
|  | fi |
|  |  |
|  | exec\_check=$(docker exec "$c" ps -el 2>/dev/null) |
|  | if [ $? -eq 255 ]; then |
|  | if [ $printcheck -eq 0 ]; then |
|  | warn "$check\_5\_7" |
|  | printcheck=1 |
|  | fi |
|  | warn " \* Docker exec fails: $c" |
|  | fail=1 |
|  | fi |
|  |  |
|  | done |
|  | # We went through all the containers and found none with sshd |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_7" |
|  | fi |
|  |  |
|  | # 5.8 |
|  | check\_5\_8="5.8 - Do not map privileged ports within containers" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | # Port format is private port -> ip: public port |
|  | ports=$(docker port "$c" | awk '{print $0}' | cut -d ':' -f2) |
|  |  |
|  | # iterate through port range (line delimited) |
|  | for port in $ports; do |
|  | if [ ! -z "$port" ] && [ "0$port" -lt 1024 ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_8" |
|  | warn " \* Privileged Port in use: $port in $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Privileged Port in use: $port in $c" |
|  | fi |
|  | fi |
|  | done |
|  | done |
|  | # We went through all the containers and found no privileged ports |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_8" |
|  | fi |
|  |  |
|  | # 5.10 |
|  | check\_5\_10="5.10 - Do not use host network mode on container" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | mode=$(docker inspect --format 'NetworkMode={{ .HostConfig.NetworkMode }}' "$c") |
|  |  |
|  | if [ "$mode" = "NetworkMode=host" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_10" |
|  | warn " \* Container running with networking mode 'host': $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Container running with networking mode 'host': $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found no Network Mode host |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_10" |
|  | fi |
|  |  |
|  | # 5.11 |
|  | check\_5\_11="5.11 - Limit memory usage for container" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | docker inspect --format '{{ .Config.Memory }}' "$c" 2> /dev/null 1>&2 |
|  |  |
|  | if [ "$?" -eq 0 ]; then |
|  | memory=$(docker inspect --format '{{ .Config.Memory }}' "$c") |
|  | else |
|  | memory=$(docker inspect --format '{{ .HostConfig.Memory }}' "$c") |
|  | fi |
|  |  |
|  | if [ "$memory" = "0" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_11" |
|  | warn " \* Container running without memory restrictions: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Container running without memory restrictions: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found no lack of Memory restrictions |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_11" |
|  | fi |
|  |  |
|  | # 5.12 |
|  | check\_5\_12="5.12 - Set container CPU priority appropriately" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | docker inspect --format '{{ .Config.CpuShares }}' "$c" 2> /dev/null 1>&2 |
|  |  |
|  | if [ "$?" -eq 0 ]; then |
|  | shares=$(docker inspect --format '{{ .Config.CpuShares }}' "$c") |
|  | else |
|  | shares=$(docker inspect --format '{{ .HostConfig.CpuShares }}' "$c") |
|  | fi |
|  |  |
|  | if [ "$shares" = "0" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_12" |
|  | warn " \* Container running without CPU restrictions: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Container running without CPU restrictions: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found no lack of CPUShare restrictions |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_12" |
|  | fi |
|  |  |
|  | # 5.13 |
|  | check\_5\_13="5.13 - Mount container's root filesystem as read only" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | read\_status=$(docker inspect --format '{{ .HostConfig.ReadonlyRootfs }}' "$c") |
|  |  |
|  | if [ "$read\_status" = "false" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_13" |
|  | warn " \* Container running with root FS mounted R/W: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Container running with root FS mounted R/W: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found no R/W FS mounts |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_13" |
|  | fi |
|  |  |
|  | # 5.14 |
|  | check\_5\_14="5.14 - Bind incoming container traffic to a specific host interface" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | for ip in $(docker port "$c" | awk '{print $3}' | cut -d ':' -f1); do |
|  | if [ "$ip" = "0.0.0.0" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_14" |
|  | warn " \* Port being bound to wildcard IP: $ip in $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Port being bound to wildcard IP: $ip in $c" |
|  | fi |
|  | fi |
|  | done |
|  | done |
|  | # We went through all the containers and found no ports bound to 0.0.0.0 |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_14" |
|  | fi |
|  |  |
|  | # 5.15 |
|  | check\_5\_15="5.15 - Do not set the 'on-failure' container restart policy to always" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | policy=$(docker inspect --format 'RestartPolicyName={{ .HostConfig.RestartPolicy.Name }}' "$c") |
|  |  |
|  | if [ "$policy" = "RestartPolicyName=always" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_15" |
|  | warn " \* Restart Policy set to always: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Restart Policy set to always: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none with restart policy always |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_15" |
|  | fi |
|  |  |
|  | # 5.16 |
|  | check\_5\_16="5.16 - Do not share the host's process namespace" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | mode=$(docker inspect --format 'PidMode={{.HostConfig.PidMode }}' "$c") |
|  |  |
|  | if [ "$mode" = "PidMode=host" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_16" |
|  | warn " \* Host PID namespace being shared with: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Host PID namespace being shared with: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none with PidMode as host |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_16" |
|  | fi |
|  |  |
|  | # 5.17 |
|  | check\_5\_17="5.17 - Do not share the host's IPC namespace" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | mode=$(docker inspect --format 'IpcMode={{.HostConfig.IpcMode }}' "$c") |
|  |  |
|  | if [ "$mode" = "IpcMode=host" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | warn "$check\_5\_17" |
|  | warn " \* Host IPC namespace being shared with: $c" |
|  | fail=1 |
|  | else |
|  | warn " \* Host IPC namespace being shared with: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none with IPCMode as host |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_17" |
|  | fi |
|  |  |
|  | # 5.18 |
|  | check\_5\_18="5.18 - Do not directly expose host devices to containers" |
|  |  |
|  | fail=0 |
|  | for c in $containers; do |
|  | devices=$(docker inspect --format 'Devices={{ .HostConfig.Devices }}' "$c") |
|  |  |
|  | if [ "$devices" != "Devices=" -a "$devices" != "Devices=[]" -a "$devices" != "Devices=<no value>" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | info "$check\_5\_18" |
|  | info " \* Container has devices exposed directly: $c" |
|  | fail=1 |
|  | else |
|  | info " \* Container has devices exposed directly: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none with devices |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_18" |
|  | fi |
|  |  |
|  | # 5.19 |
|  | check\_5\_19="5.19 - Override default ulimit at runtime only if needed" |
|  |  |
|  | # List all the running containers, ouput their ID and host devices |
|  | fail=0 |
|  | for c in $containers; do |
|  | ulimits=$(docker inspect --format 'Ulimits={{ .HostConfig.Ulimits }}' "$c") |
|  |  |
|  | if [ "$ulimits" = "Ulimits=" -o "$ulimits" = "Ulimits=[]" -o "$ulimits" = "Ulimits=<no value>" ]; then |
|  | # If it's the first container, fail the test |
|  | if [ $fail -eq 0 ]; then |
|  | info "$check\_5\_19" |
|  | info " \* Container no default ulimit override: $c" |
|  | fail=1 |
|  | else |
|  | info " \* Container no default ulimit override: $c" |
|  | fi |
|  | fi |
|  | done |
|  | # We went through all the containers and found none without Ulimits |
|  | if [ $fail -eq 0 ]; then |
|  | pass "$check\_5\_19" |
|  | fi |
|  | fi |