Part 1 - Buffer Overflow Lab Discussion Questions

Answer the following questions regarding the Buffer Overflow Lab in this module.

1. How did you figure out the required length before overflow?

I Started with 10 characters, added increments of 5 till I hit the overflow than deleted in increments of 1 charcaters to find the answer of 16.

2. What were the likely culprits in the buffer overflow?

no limits or constraints in the password field

3. How should you design around these issues to make programs less susceptible to these failures?

while writting the code for Password verification, Validate the user input by limiting the charcaters to a set amount, if the user goes over the limit yeild a return with a qoute stating:

"You have entered the wrong Password" Than Break out of the Loop.

4. Describe a policy that you can put in place for software development to mitigate the possibility of such issues making it into the final program code.

Three possible policies to implement are:

\* Storing only salted cryptographic hashes of passwords and never storing plain-text passwords.

\* Enforcing password length and complexity requirements.

\*Disable password entry after multiple incorrect login attempts.

Part 2 – Firewall Rule Creation

Scenario: You are in charge of managing the corporate firewall for your hospital branch. You will need to both create and verify that you have implemented correct firewall policy in the following examples.

System 1: You need to set up a firewall for an internet-facing web server. The server must be able to display both HTTP and HTTPS pages to customers outside of the network. All other traffic should be blocked. The server also runs a local database system like MariaDB. However, all connections occur locally within the machine itself. Lastly, you’ll want SSH access from within your on-site network (192.168.1.0/24). Because it’s a Linux server, you will be using IPTables to write the rules. Please write them for this configuration in the correct order for the input chain.

# Firewall rules for HTTP & HTTPS

sudo iptables -P INPUT ACCEPT

sudo iptables -P OUTPUT ACCEPT

sudo iptables -P FORWARD ACCEPT

sudo iptables -F

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

sudo iptables -A OUTPUT -p tcp -j ACCEPT

sudo iptables -I INPUT 1 -i lo -j ACCEPT

# Drop all other Policies

sudo iptables -P INPUT DROP

sudo iptables -P OUTPUT DROP

sudo iptables -P FORWARD DROP

# open MariaBD port

sudo ufw allow 3306

# only allow subnet 192.168.1.0/24 to connect to our MariaBD server

sudo ufw allow from 192.168.1.0/24 to any port 3306

# Add SSH access to a subnetwork

sudo iptables -A INPUT -p tcp --dport 22 --source 192.168.1.0/24 -j ACCEPT

# Drop all other sources

sudo iptables -A INPUT -p tcp --dport 22 -j DROP

System 2: A customer wants to use a virtual private network to access their work computer from home. The remote system uses port 51820 as its VPN listening port. Write firewall rules for BOTH the home (remote) unit and the work system which connects to the home user’s VPN. Also, describe potential issues that the customer may face.

Customer's home network is vulnerable to attacks and can cause a weak link to the Work computers netwok.

\*After setting up certificates and Generated Keys for Server VPN and coping the .config file to the Client VPN:

Client VPN: IF=10.0.2.8 Server VPN: IF= 10.0.2.7

VPNServer="10.0.2.7" sudo iptables -I INPUT -p udp --dport 51820 -j ACCEPT

wan="eth0" sudo iptables -t nat -A POSTROUTING -s 10.0.2.0/24 -o eth0 -j MASQUERADE

lan="eth1" sudo iptables-save > /etc/sysconfig/iptables

tun="tun0" sudo sysctl -w net.ipv4.ip\_forward=1

# Flush rules

/sbin/iptables -F

/sbin/iptables -F -t nat

# Enable NAT

/sbin/iptables -t nat -A POSTROUTING -o $tun -j MASQUERADE

/sbin/iptables -t nat -A POSTROUTING -o $wan -j MASQUERADE # Needed to SSH to VPN server

# Allow SSH to the VPN server itself

/sbin/iptables -A FORWARD -o $wan --destination $VPNServer --protocol tcp --dport 22 -j ACCEPT

/sbin/iptables -A FORWARD -i $wan --source $VPNServer --protocol tcp --sport 22 -j ACCEPT

# Allow VPN traffic

/sbin/iptables -A FORWARD -i $lan --destination $VPNServer --protocol udp --dport 51820 -o $tun -j ACCEPT

/sbin/iptables -A FORWARD -i $tun --source $VPNServer --protocol udp --sport 51820 -o $lan -j ACCEPT

# Block non-VPN traffic across the WAN (Internet) interface (after VPN setup)

/sbin/iptables -A FORWARD -i $wan -j DROP

/sbin/iptables -A FORWARD -o $wan -j DROP

# Allow VPN client to connect to VPN server

/sbin/iptables -A INPUT -i $wan --source $VPNServer --protocol udp --sport 51820 -j ACCEPT

/sbin/iptables -A OUTPUT -o $wan --destination $VPNServer --protocol udp --dport 51820 -j ACCEPT

System 3: A networked embedded system serves a basic webserver over HTTPS, and listens for mDNS and UPnP broadcasts on a local network. Secondly, ingress filtering (input chain) egress filtering (the output chain) must be implemented such that the device ONLY accepts requests from within the local network subnet. The subnet is found at 192.168.1.0/24. Write the firewall rules to enable each of these services. You may need to read and look up each of those services to understand the relevant ports.

# Firewall rules for HTTPS

sudo iptables -P INPUT ACCEPT

sudo iptables -P OUTPUT ACCEPT

sudo iptables -P FORWARD ACCEPT

sudo iptables -F

sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

sudo iptables -A OUTPUT -p tcp -j ACCEPT

sudo iptables -I INPUT 1 -i lo -j ACCEPT

# Drop all other Policies

sudo iptables -P INPUT DROP

sudo iptables -P OUTPUT DROP

sudo iptables -P FORWARD DROP

# open mDNS port

sudo ufw allow 5353

# open upnp port only on LAN

iptables -A INPUT -s 192.168.1.0/24 -m tcp -p tcp --dport 49200 -j ACCEPT

iptables -A INPUT -s 192.168.1.0/24 -m udp -p udp --dport 49200 -j ACCEPT

iptables -A INPUT -s 192.168.1.0/24 -m udp -p udp --dport 1900 -j ACCEPT

# only allow subnet 192.168.1.0/24 to connect to embedded system

sudo ufw allow from 192.168.1.0/24 to any port 5353

# Add SSH access to a subnetwork

sudo iptables -A INPUT -p tcp --dport 22 --source 192.168.1.0/24 -j ACCEPT

# Drop all other sources

sudo iptables -A INPUT -p tcp --dport 22 -j DROP