Chad Ballay

**Week 4 – Routing Fundamentals**

1. Provide the definition for Access Control Lists (ACL) and how they are used on servers and on a network.  Explain any similarities and differences in how ACLs are used. Research and describe at least two outside references that discusses ACLs and their use for security.
2. Provide an example of how a packet travels from a host computer and is routed by a router. You should provide a basic overview of what steps are involved in the routing of the packet to its final destination.

**Access Control Lists**

Network Access Control Lists(ACLs) are rules that enforce how the flow of incoming and outgoing traffic is filtered. The defacto standard comes from CISCO based implementations so much of the details model after their technical leadership. At the basic functionality the rules are set to inspect each packet and then based on criteria defined in the ACL, to either allow or deny the packet from continuing on. The criteria that can be used is protocol (UDP/TCP/ICMP), source IP, destination IP, port number. Each rule is applied in turn so the cumulative effect of all the rules dictates the state of the filtering. (Messer, n.d.)

Network ACL Example (Configure Commonly Used IP ACLs, n.d.)

A screenshot of a social media post

Description automatically generated

File system ACLs perform a similar role. In these ACLs you define the permissible behavior but instead of on networking behavior these apply to interactions on file objects. For example you could define that a user has read/write access to a directory, or no users but an explicitly enumerated set have read access to a file.

Filesystem ACL Example (Cánepa, n.d.)

A screenshot of a cell phone

Description automatically generated

So both Network and Filesystem ACL control the access behavior they govern. Can this traffic flow from here to there? Can this user delete this file? The difference is the type of access they govern. Network traffic vs the manipulation of a filesystem object. This makes them exceptionally powerful for securing a system and easy to get wrong. For ACL’s each rule is interpreted in turn so by the end of processing each rule there is a good chance of redundantly defined behavior. Several third party applications exist to find these gaps or duplications.

* Cisco-ACL-Analyzer <https://github.com/frenzymadness/Cisco-ACL-Analyzer>
* Network MOM ACL Analyzer <https://www.youtube.com/watch?v=KITTaPnSx_c>
* SolarWinds Permission Analyzer <https://www.solarwinds.com/free-tools/permissions-analyzer-for-active-directory>

**Life of a Packet**

This is a rough summation taken from an indepth example provided by Nicolas Mesa. (Mesa, n.d.)

In the beginning there is the kernel call for a DNS query. Google Chrome has let the kernel know that it wishes to find the IP of www.chadballay.com. It knows it’s own IP address so the SourceIP is filled in as is the DestPort since it is DNS so that becomes 53. The DestinationIP is set the DNS server.

Onto the Ethernet part. The SourceMac is gathered from the local machine. The DestinationMac needs to be the local gateway’s MAC since when the computer looks up the DNS server’s ip address it sees that it isn’t local to this network. So it looks in its local ARP cache and puts the gateways MAC address in as the DestinationMac.

So our packet is roughly done and the computer sends it to the gateway’s MAC address. The gateway(router) receives this packet. Subtracts 1 from the TTL. It also notes that the DestinationIP is outside of it’s network. (BGP and some other items occur here.) It calculates/updates both the Destination and Source MAC addresses and hands the packet off to the next hop in the path. Repeat this process into you get to your final router that recognizes the Destination IP address. That receiving computer gets the packet, unwraps the layers and process the DNS query.

This whole process is walked backwards in a similar manner with the results of the DNS query. The data is wrapped in the protocol layer. An IP address for the source and destination is gathered. The SourceMac is filled in and the next hops DestinationMac is calculated. The packet then traverses the network back being handed from one gateway to the next until it reaches the original sender.

Observations:

* The route to and from one endpoint doesn’t need to be the same. Ideally it should always end up being the most direct but there are often times where the path will vary.
* Cache is king. The neutered example up above is just for the DNS query. When I load Bellevue.edu a 106 requests are made when I look at the network tab in Developer Tools. 1.4 MB worth data. DNS cache, browser cache, etc…. It all matters and it all is useful. Within reason.
* Wireshark is something I need to spend more time with. While trying to find a packet to look at, I noticed a whole mess of traffic going on with my wireless card set to promiscuous mode.