# Post 1

Thursday, January 24, 2019 11:28 PM

MODERN NETSEC THREATS CH.1

**Cisco SIO: Security Intelligence Operations:** Provides alerts regarding current network attacks

- Updates in real time/provides info to help ID threats

| | |
|---|---|
| **Vuln** | Weakness/flaw in network: Can be exploited<br>Weak/unsecure protocols/config errors/weak sec policy |
| **Threat** | Potential for vuln to turn into attack [Malware/exploits/etc…] |
| **Mitigation** | Reducing severity of a vuln |
| **Risk** | Potential of threat to exploit vulns: Measured using probability of occurrence |

**Vectors:** Path/means an attacker can use to gain access to server/host/network
**Data exfiltration:** Data intentionally/unintentionally lost/stolen/leaked
**DLP: Data Loss Prevention:** Controls should be implemented (strategic/op/tactical measures)
**Vectors of Data Loss:**

| | |
|---|---|
| **Email/Social Networking** | Instant msging social media: Can captured/reveal info |
| **Improper Access Control** | Stolen/weak passwds can provide easy access to data |
| **Hard Copy** | Corporate data should be disposed of thoroughly (dumpster diving) |
| **Removable Media** | Unauthorized data transfers to media like USB drives |
| **Cloud Storage Devices** | Sensitive data can be lost if access to cloud is compromised |
| **Unencrypted Devices** | If data isn't encrypted: It's open |

**CAN: Campus Area Network:** Interconnected LANs w/in limited geographic area
**WAN: Wide Area Network:** Span a wide geographic area: ASA/VPN
**Data Centers:** Typically off-site facility: Stores sensitive/proprietary data
**Physical sec can be divided into 2 areas:**

1. **Outside perimeter:** On premise officers/fences/gates/video/alarms
2. **Inside perimeter:** Video/motion detectors/sec traps/biometric access/exit sensors

| | |
|---|---|
| **Security traps** | Mantrap: Provides access to data halls: Similar to air lock<br>• Uses badge ID card: Facial recognition/fingerprints/biometrics |

**Cloud/Virtual Networks:** Virtualization foundation of Cloud/Separates app/OS from HW
**Cloud:** Physical/virtual servers commonly housed in data centers: VM's provide services to clients
**VMs prone to specific attacks:**

| | |
|---|---|
| **Hyperjacking** | Hijacking hypervisor (controller SW)/using as launch point |
| **Instant On Action** | When VM hasn't been used for a time is brought online: May be outdated |
| **AV Storms** | All VMs attempt to DL AV files at same time |

**Core components of Cisco Secure Data Center provide:** Sec Segmentation/Threat Defense/Visibility
**MDM: Mobile Device Management features:**

| | |
|---|---|
| **Data Encryption** | MDM supports encryption to access network/corporate content |
| **PIN Enforcement** | Effective in preventing unauth access to device<br>• Passwd policies can be enforced, reducing likelihood of brute-force |
| **Data Wipe** | Lost/stolen devices can be remotely fully/partially-wiped: By usr/admin |
| **DLP** | **Data Loss Prevention:** Data protection functions prevent unauth usrs from accessing data<br>• DLP prevents auth usrs from doing careless/malicious things w/it |
| **Jailbreak/Root Detection** | **Jailbreaking (iOS)/rooting (Android):** Can detect bypasses/restrict device access to network/assets |

**Hacking:** 1960's: Phone phreaking: Various audio freq could manipulate phone sys

- Tone dialing (switches used various tones to indicate diff functions)
- Early hackers realized mimicking tone w/whistle (CC/2600) could exploit switches to make free long-distance calls

**Mid-1980s:** Dial-up modems used to connect computers to Internet
**War dialing programs:** Dialed each number in a given area searching for computers, BBS, fax machines
**Evolution of Sec Tools:**

| | |
|---|---|
| **Passwd Crackers** | Cracking tools often used to recover/crack passwds: Done by either:<br>• Removing original passwd after bypassing encryption<br>• Discovery of passwd<br>**Examples:**<br>• John the Ripper<br>• Ophcrack<br>• L0phtCrack<br>• THC Hydra<br>• RainbowCrack<br>• Medusa |
| **Wireless** | Tools to hack into wireless networks<br>**Examples:**<br>• Aircrack-ng<br>• Kismet<br>• InSSIDer<br>• KisMAC<br>• Firesheep<br>• NetStumbler |
| **Network Scanning** | Probes network devices/services/hosts for open TCP/UDP ports<br>**Examples:**<br>• Nmap<br>• SuperScan<br>• Angry IP Scanner<br>• NetScanTools |
| **Packet Crafting** | Probe/test firewall's str using specially forged packets<br>**Examples:**<br>• Hping<br>• Scapy<br>• Socat<br>• Yersinia<br>• Netcat<br>• Nping<br>• Nemesis |
| **Packet Sniffers** | Captures/analyzes packets w/in traditional LANs/WLANs<br>**Examples:**<br>• Wireshark<br>• Tcpdump<br>• Ettercap<br>• Dsniff<br>• EtherApe<br>• Paros<br>• Fiddler<br>• Ratproxy<br>• SSLstrip |
| **Rootkit Detectors** | Dir/file integrity checker to detect installed root kits<br>**Examples:**<br>• AIDE<br>• Netfilter<br>• OpenBSD Packet Filter |
| **Fuzzers** | Discover sys sec/vulns |

| | |
|---|---|
| | **Examples:**<br>• Skipfish<br>• Wapiti<br>• W3af |
| **Forensic** | Sniff out any trace evidence existing in particular sys<br>**Examples:**<br>• Sleuth Kit<br>• Helix<br>• Maltego<br>• Encase |
| **Debuggers** | RE binary files when writing exploits: Also for analyzing malware<br>**Examples:**<br>• GDB<br>• WinDbg<br>• IDA Pro<br>• Immunity Debugger |
| **OS's** | OS's w/tools/tech for hacking<br>**Examples:**<br>• Kali<br>• SELinux<br>• Knoppix<br>• BackBox |
| **Encryption** | Safeguard contents: Tools use algs to encode data to prevent unauth access<br>**Examples:**<br>• VeraCrypt<br>• CipherShed<br>• OpenSSH<br>• OpenSSL<br>• Tor<br>• OpenVPN<br>• Stunnel |
| **Vuln Exploitation** | ID whether remote host is vuln to attack<br>**Examples:**<br>• Metasploit<br>• Core Impact<br>• Sqlmap<br>• Social Engineer Toolkit<br>• Netsparker |
| **Vuln Scanners** | Scan network/sys to ID open ports: Can scan for vulns/VM's/BYOD devices/DB's<br>**Examples:**<br>• Nipper<br>• Secunia PSI<br>• Core Impact<br>• Nessus<br>• SAINT<br>• Open VAS |

## Categories of Attack Tools:

| | |
|---|---|
| **Eavesdropping** | Capturing/listening to network traffic: AKA sniffing/snooping |
| **Data Modification** | Altering captured traffic in packets w/out knowledge of sender/receiver |
| **IP Spoofing** | Construction of packet that appears to originate from valid address |
| **Passwd-Based** | |
| **DoS** | Denial-of-Service: Crashes apps/services/floods a machine w/traffic/blocks it overloading everything |

| | |
|---|---|
| **MiTM** | Positioning yourself bet source/destination: Actively monitor/capture/control comm transparently |
| **Compromised-Key** | Obtaining secret key: Can be used to gain access to secured comm w/out sender/receiver knowing |
| **Sniffer** | App/device: Can read/monitor/capture data exchanges/packets<br>• Even encapsulated (tunneled) packets can be broken open/read |

**Types of Malware:** End devices: Prone to attacks
**Virus:** Code attached to exe files/often legitimate programs. Most require end user activation/can lay dormant
**Trojan Horse:** From Greek mythology: Warriors offered ppl of Troy giant horse: Attacked Trojans in sleep

- Malware carries ops under guise of desired function: Code exploits privs of usr that runs it

**Trojan Classification:**

| | |
|---|---|
| **Remote-access** | Enables unauth access |
| **Data-sending** | Provides attacker w/sensitive data (passwds) |
| **Destructive** | Corrupts/del files |
| **Proxy** | Uses victim's machine as source device to launch attacks |
| **FTP** | Enables unauth FTP services on end devices |
| **Sec SW disabler** | Stops AV's/FW's from functioning |
| **DoS** | Slows/Halts network activity |

**Worms:** Replicates self independently exploiting vulns: Usually slows down networks

- Virus requires host to run: Worm doesn't

| | |
|---|---|
| **Code Red** | **2001**: Infected 658 servers: W/in 19 hours? Infected over 300,000 |
| **SQL Slammer** | "***The worm that ate the Internet***" DoS that exploited a buffer overflow in SQL Server<br>• Peak: Doubled in size every 8.5s<br>• Infected 250,000+ hosts w/in 30 min<br>• **Jan 25, 2003**: Disrupted Internet/financial institutions/ATM's/etc…<br>Patch for vuln released 6 months earlier |
| **MyDoom** | **2004**: Activated by unsuspecting usr (User1) by opening attachment<br>• Released worm was able to learn all avail emails on sys<br>• Would then send spam to all recipients discovered |
| **Conficker** | **2008**: Next largest worm since SQL Slammer |

**Worm Components:**

| | |
|---|---|
| **Enabling vuln** | Installs itself using exploit mech (email/exe/Trojan) on vuln sys |
| **Propagation mechanism** | After gaining access: Replicates itself/locates new targets |
| **Payload** | Malicious code results in some action: Usually used to create backdoors to infected hosts/DoS attacks |

**Other Malware:** Examples of modern malware:

| | |
|---|---|
| **Ransomware** | Denies access to sys: Demands ransom for restriction removal |
| **Spyware** | Gathers info about usr/sends to 3rd entity w/out consent |
| **Adware** | Annoying pop-ups generate revenue |
| **Scareware** | Scam SW uses social eng to shock/induce anxiety by creating perception of threat |
| **Phishing** | Attempts to convince people to divulge sensitive info |
| **Rootkits** | Installed on compromised sys. After: Continues to hide intrusion/maintain priv access |

**Network Attacks:** Malware is means to get a payload delivered

| | | |
|---|---|---|
| Reconnaissance | Access | DoS |

**Reconnaissance:** Info gathering: Example: Thieves surveying neighborhood

| | |
|---|---|
| **Info query** | Looking for info about target: OSINT/Google/orgs website/whois/etc… |

| | |
|---|---|
| **Ping sweep** | Determines which IP's active |
| **Port scan** | Determines which ports/services avail |
| **Vuln Scanners** | Querying ID'd ports to determine type/vers of app/OS running on target |
| **Exploitation** | Attempts to discover vuln services that can be exploited |

**Access Attacks:** Exploits known vulns in auth/FTP/web services: Entry to web accts, DB's/sensitive info
**At least 3 reasons for access attacks:** Retrieve data/gain access/escalate privs
**Types of Access: 5 common types of access attacks:**

| | |
|---|---|
| **Passwd attack** | Discovering critical sys passwds using various methods (social/dictionary/brute-force) |
| **Trust exploitation** | Unauth privs to gain access to sys/possibly compromising target |
| **Port redirection** | Using compromised sys as base for attack against other targets |
| **MiTM** | Being positioned bet 2 legitimate entities in order to read/mod data that passes bet them |
| **Buffer overflow** | Exploiting buffer mem/overwhelming it w/unexpected values (dislike this def) |
| **IP, MAC, DHCP Spoofing** | Device attempts to pose as another by falsifying data: Multiple types<br>• MAC spoofing occurs when 1 machine accepts data packets based on MAC of another machine |

**Social Engineering:** Attempts to manipulate individuals into performing actions: Relies on the 'goodness' of people
**Types of social engineering:**

| | |
|---|---|
| **Pretexting** | Calling an individual/lying to them in an attempt to gain access to data |
| **Phishing** | When a malicious party sends fraudulent email disguised as legitimate |
| **Spear phishing** | Targeted phishing attack tailored for specific individual/org |
| **Spam** | |
| **Tailgating** | When someone quickly follows an auth person into sec location |
| **Something for Something (Quid pro quo)** | When personal info from a party is exchanged for something like a free gift |
| **Baiting** | When malware-infected devices (USB drive) are left to be found on purpose |
| **Social Engineering Toolkit** | Designed to help WH's create social attacks to test their own networks |

**DoS: Denial of Service Attacks:** Highly publicized network attacks. Result some sort of interruption of service
**2 more sources:**

| | |
|---|---|
| **Maliciously Formatted Packets** | When fwded to host/app: Receiver is unable to handle: Can cause crash/slow down things |
| **Overwhelming Quantity of Traffic** | When network/host/app unable to handle enormous quantity of data: Sys crashes/goes slow |

**3 DoS attacks introduced for historical reasons:**

| | |
|---|---|
| **Ping of Death** | Legacy: An echo request in a packet larger than max size of 65,535 bytes<br>• Receiving host wouldn't be able to handle packet that size: Crash |
| **Smurf Attack** | Legacy: Large # of ICMP requests were sent to various recipients: Using multiple recipients amplified it<br>• Packet source address contained spoofed IP of intended target<br>• Type of reflection attack b/c echo replies would be reflected back to targeted host to overwhelm<br>Mitigated w/**no ip directed-broadcast** cmd: Default int setting as of IOS 12<br>• Reflection/amplification technique continues to be used in newer forms of attacks |
| **TCP SYN Flood Attack** | Usr sends TCP SYN request packets w/spoofed source address to target<br>• Target replies w/TCP SYN-ACK packet to spoofed address/waits for a TCP ACK packet<br>• Responses never arrive: Target hosts overwhelmed w/TCP half-open connections |

**DDoS Attacks:** Distributed: Similar in intent to DoS, except increases in magnitude b/c of multiple

sources
**Botnet:** Network of infected hosts
**Zombies:** Compromised computers: Continue to scan/infect more targets to create more zombies
**Handler:** When ready, handler makes botnet of zombies to carry out DDoS
**CIA: Confidentiality/Integrity/Availability:** Deals w/protecting info/sys
Network Security Domains: Vital to understand reasons for netsec
12 netsec domains: Domains provide a framework for discussion

- Specified by **ISO: International Org for Standardization || IEC: International Electrotechnical Commission**
- **ISO/IEC 27002:** Serve to organize at a high lvl, the vast realm of info under netsec
- Intended to serve as a common basis for developing org sec standards/sec mgmt practices

**NetSec Policy:** Broad, end-to-end doc designed to be applicable to an org's ops

- Used to aid in design/sec principles/deployments

**Policy Objectives:** Encompasses all reqs for securing resources: Not just equip reqs/procedures
**Security Artichoke**

In this analogy, hackers no longer have to peel away each layer: They only need to remove certain 'artichoke leafs'.

- Heart of the artichoke is where the most confidential data is found
- Each leaf provides a layer of protection while simultaneously providing path to attack

**Evolution of NetSec Tools:** In the 90s, security became an integral part of everyday operations
**IDS: Intrusion Detection System:** One of the 1st netsec tools

- IDS and now **IPS: Intrusion Prevention System** provide real-time detection of certain types of attacks
- Unlike an IDS, an IPS can also auto block in real-time

Another device developed was the firewall:

- Designed to prevent undesirable traffic from entering prescribed areas w/in a network, thereby providing perimeter sec
- Original firewalls were SW features added to existing devices

**SecureX Sec Tech:** Cisco SecureX arch designed to provide effective sec for any usr/using any device/from any location/any time

- New arch uses higher-lvl policy lang that takes into acct full context of a situation [who, what, where, when, how]
- Highly distributed sec policy enforcement, security is pushed closer to where end user is working

**Includes 5 major components:**

1. Scanning Engines
2. Delivery Mechanisms
3. Security Intelligence Operations (SIO)
4. Policy Management Consoles
5. Next-Generation Endpoints

**Centralized Context-Aware Network Scanning Element:**

- A device that examines packets on the wire, but also looks at external info to understand the full context
- To be context aware, the scanner must consider the 'who, what, where, when, and how' of a packet as it relates to sec.

**A context-aware policy uses 5 params:**

1. Person's identity
2. App in use
3. Type of device being used for access
4. Location
5. Time of access

**Cisco Sec Intel Ops:** A Cloud-based service that connects global threat info, reputation-based services, and analysis, to Cisco devices
**SIO uses a variety of sources when ID/categorizing threats:**

- Blacklist/rep filters
- Info gathered by spam traps, honeypots, crawlers
- ID/registering valid site domains
- Known DB of attack sigs
- Content inspection
- 3rd party partnerships

**Mitigating Malware:** Primary means of mitigating virus and Trojan horse attacks is AV SW: Host-based.
**Mitigating Worms:** Worms are more network-based:

Can be broken down into 4 phases:

1. Containment
2. Inoculation
3. Quarantine
4. Treatment

**Mitigating Reconnaissance:** Preconfigured alarms: Triggered when certain params exceeded (# of ICMP reqs per s)

1. Anti-sniffer SW/HW: Detect changes in response time of hosts to see if hosts processing more traffic than loads indicate
2. Encryption is also effective for mitigating packet sniffer attacks
3. Using an IPS/FW can limit info discovered w/a port scanner
4. Ping sweeps can be stopped if ICMP echo/echo-reply are turned off on edge routers (when off diagnostics are lost)

**Mitigating Access:** Strong passwds/disable accts after # of unsuccessful logins/encrypted/hashed auth protocols/education

**Mitigating DoS:** Utilization SW package should be running at all times/antispoofing tech

- Port sec/DHCP snooping/IP source guard/ACL's/ARP inspection

**NFP Framework: Cisco Network Foundation Protection framework:**

- Provides comprehensive guidelines for protecting infrastructure

**NFP logically divides routers/switches into 3 planes:**

| | |
|---|---|
| **Control** | Routing data correctly: Consists of device-generated packets required for op of network<br>• ARP msg exchanges/OSPF routing advertisements |
| **Mgmt** | Responsible for managing elements. Generated by devices/mgmt stations using processes/protocols<br>• Telnet/SSH/TFTP/FTP/NTP/AAA/SNMP/syslog/TACACS+/RADIUS/NetFlow |
| **Data(Fwding)** | Fwding data: Traffic normally consists of user-generated packets being fwded bet end devices<br>• Most traffic travels through router/switch via data plane |

**Securing Control Plane:**

| | |
|---|---|
| **Routing protocol auth** | R-protocol auth/neighbor auth: Prevents rtr from accepting fraudulent updates |
| **Control Plane Policing** | CoPP: Cisco IOS feature: Allows users to control flow of traffic handled by route processor of device |
| **AutoSecure** | Can lock down mgmt plane functions/fwding plane services/functions of a router |

**Securing Mgmt Plane:**

| | |
|---|---|
| **Login/passwd policy** | Restricts device accessibility: Limits accessible ports/restricts who/how methods of access |
| **Legal notification** | Legal notices |
| **Confidentiality of data** | Protect locally stored data from being viewed/copied: Strong auth |
| **Role-based (RBAC)** | Access only granted to auth usrs/groups/services: RBAC/AAA |
| **Auth actions** | Restricts actions/views permitted by any usr/group/service |
| **Enable mgmt access reporting** | Logs/accts for all access |

**Securing Data Plane:**

| | |
|---|---|
| Block unwanted traffic/usrs | ACLs can filter incoming/outgoing packets on an int |
| Reduce chance of DoS | ACLs can be used: TCP intercept can be config |
| Mitigating spoofing attacks | ACLs to mitigate spoofing attacks |
| Providing BW control | ACLs on a slow link can prevent excess traffic |
| Classifying traffic to protect Mgmt/Control planes | ACLs can be applied on vty lines |

**L2 sec tools integrated into Cisco Catalyst switches:**

| | |
|---|---|
| **Port security** | Prevents MAC spoofing/flooding |
| **DHCP snooping** | Prevents client attacks on DHCP server/switch |

| | |
|---|---|
| **Dynamic ARP Inspection (DAI)** | Sec to ARP using DHCP snooping table to min impact of ARP poisoning/spoofing |
| **IP Source Guard** | Prevents spoofing addresses by using DHCP snooping table |

# Post 2

Thursday, January 24, 2019 11:30 PM

SECURING NETWORK DEVICES P1

**Device hardening**: Critical task: Appropriate sec policies/controls must be implemented:
**Routers:** Primary target b/c they affect/direct and get into/out of traffic bet networks
**Edge Router Sec Approaches**

| | |
|---|---|
| **Single Router** | Connects protected network/LAN to Internet: Smaller sites: SOHO:<br>• Can be supported by ISR's: Integrated Service Routers<br>• Doesn't impede performance |
| **Defense-in-depth** | Multiple layers of sec prior to traffic entering LAN: Better than single<br>**3 primary layers of defense:**<br>**1. Edge router:** 1st line defense: **AKA Screening router**<br>○ After traffic filtering: Passes all connections intended for LAN to FW<br>**2. Firewall:** Picks up where Edge left: Additional filtering<br>○ Additional access control: Tracks state of connections: Checkpoint device<br>○ Default: Denies connections from outside/untrusted to inside<br>○ Allows internal usrs: Establish connections to untrusted<br>○ Permits responses come back through FW<br>○ Usr auth to gain access to resources<br>**3. Internal router**: Sec tools: IPS's/web sec apps/email sec apps |
| **DMZ** | **Demilitarized Zone:** Variation of defense-in-depth: Includes intermediate area:<br>• Servers that must be accessible from Internet/external<br>**Can be set bet 2 routers:**<br>**Internal router:** Connecting to protected network<br>**External router:** Connecting to unprotected<br>**Can be an additional port off single router:**<br>• FW: Located bet protected/unprotected<br>Permit required connections [HTTP] from outside to public servers in DMZ<br>• FW: Primary protection for all devices in DMZ |

**Areas of Router Security**
1. Physical Sec
2. Operating System Sec

| | |
|---|---|
| **Physical** | Router/Phys devices connected in a secure locked room? Only accessible to auth personnel<br>• Free of electrostatic/magnetic interference<br>• Fire suppression<br>• Temp/humidity controls<br>• UPS: Uninterruptible PS: Diesel backup power generator<br>• Spare components avail<br>Reduces possibility of outage from power loss |
| **OS** | Config: Max amt of mem: Helps mitigate some DoS attacks: Supports widest range of sec services<br>• Latest stable ver of OS: Meets specs of devices<br>• Sec/encryption in OS: Improved/updated over time<br>• Secure copy of OS imgs/rtr config files as backups |

**Hardening: Eliminate potential abuse of unused ports/services:**
- Admin control: Ensure only auth personnel have access: Controlled
- Disable unused ports/ints: Reduce # of ways device can be accessed
- Disable unnecessary services

**Impt tasks: Access to device:**

| | |
|---|---|
| **Restrict accessibility** | Limit accessible ports: Restrict perm comm: Restrict methods of access |
| **Log/Acct for all** | Record anyone who accesses device: What/When during access: Auditing purposes |

| | |
|---|---|
| **access** | |
| **Auth access** | Ensure access granted only to auth usrs/groups/services: Limit # of failed logins/time allowed bet logins |
| **Auth actions** | Restrict actions/views perm by any usr/groups/service |
| **Legal notification** | Display legal notice: Dev w/company counsel: Interactive sessions |
| **Ensure confidentiality** | Protect locally stored/sensitive data from being viewed/cp:<br>• Vuln of data in transit: Over comm chan: Sniffing/session hijacking/MITM attacks |

**Local/Remote Access:** Rtr can be accessed for admin purposes:

| | |
|---|---|
| **Local** | All network devices can be accessed locally<br>• Local access: Usually requires direct connection to console port on rtr/term SW<br>• Admin: Phys access to rtr/console cable to connect: Local: Initial config |
| **Remote** | Aux port avail: More common to just Telnet/SSH/HTTP/HTTPS/SNMP<br>• Local/remote network |

**Precautions: Remote:** Encrypt all traffic bet admin computer/router:
- Instead of Telnet: SSH || HTTPS instead of HTTP
- Dedicated mgmt network: Include only ID'd admin hosts/connections to dedicated int
- Packet filter: Allow only ID'd admin/hosts/preferred protocols to access

Example: Only SSH requests from IP of admin to initiate connections to rtrs
- VPN: Local network before connected to router mgmt int

**Passwds:** Ensure strong passwds used
- Cisco/many sys: Passwd-leading spaces ignored but spaces after 1st char not ignored

**Access Sec:** Default: Passwd length min 6 chars
**Increase min length: R1(config)# security passwords min-length length** *[global]*
- Default: w/exception of passwd generated by **enable secret**: Cisco passwds stored plaintxt in router start/run config files

**Encrypt all plaintxt passwds: R1(config)# service password-encryption** *[global]*

**Disable unattended connections:** Default: Admin int stays active/logged for 10 min after last session activity
**R1(config-line)# exec-timeout minutes seconds** *[line]*

**Disable EXEC process for specific line (EX: Aux port): R1(config-line)# no exec**
- Only outgoing connection on line: Disables EXEC process for connections that may attempt to send data to rtr

**Secret Passwd Algs**
**Enable Secret Defaults to MD5:**
**R1(config)# enable secret cisco12345**
**R1(config)# do show run | include enable**
enable secret 5 $1$cam7$99EfzkvmJ5h1gEbryLVRy.
**R1(config)# enable secret ?**

| | |
|---|---|
| **0** | UNENCRYPTED passwd |
| **5** | MD5 HASHED secret |
| **8** | PBKDF2 HASHED secret |
| **9** | SCRYPT HASHED secret |
| **LINE** | UNENCRYPTED enable secret |
| **Level** | Set exec lvl passwd |

**Specifying Type 9 requires an encrypted passwd**
**R1(config)# enable secret 9 cisco12345**
ERROR: The secret you entered isn't valid….
**R1(config)# enable secret 9**
$1$cam7$99EfzkvmJ5h1gEbryLVRy$1$cam7$99EfzkvmJ5h1gEbryLVRy.

**enable algorithm-type** [**md5 | scrypt | sha256**] **secret unencrypted-passwd**

| | |
|---|---|
| **md5** | **Type 5:** Message Digest Alg 5 |

| | |
|---|---|
| **scrypt** | **Type 9:** script |
| **sha256** | **Type 8:** Passwd-Based Key Derivation Function 2 w/Hash alg SHA-256 |

**Syntax:**
**R1(config)# username Esther secret cisco12345**
**R1(config)# do show run | include username**
**R1(config)# username Esther algorithm-type script secret cisco12345**

**MD5 hashes:** Not secure: Attackers can reconstruct valid certs
- Allows attackers to spoof any site
- **enable secret password** MD5 hash by default
- Config all secret passwds using Type 8/9 passwds (can't with 1841's BTW)
  - Type 8/9: Introduced in IOS 15.3(3)M | uses SHA
- Must paste encrypted passwd: Can be copied

**Enter unencrypted passwd: enable algorithm-type**
Introduced IOS 15.3(3)M: **username secret**
- Similar to enable secret : Enter usr w/username secret command
- Default: Still MD5
- **username name algorithm-type** To specify type 9
- Backwards compatibility: **enable password/username password/line password** cmds avail in IOS
- NO encryption by default: At BEST: Type 7

**Securing Line Access**
**R1(config)# username Esther algorithm-type scrypt secret cisco12345**
**R1(config)# line con 0**
**R1(config-line)# login local**
**R1(config-line)# exit**
**R1(config)# line aux 0**
**R1(config-line)# login local**
**R1(config-line)# exit**
**R1(config)# line vty 0 4**
**R1(config-line)# login local**
**R1(config-line)# transport input ssh**

**DEFAULT: con/aux ports don't require passwd for admin access**
- Passwd cmd config on console/vty/aux lines can ONLY use type 7
- Config con/aux lines for usrname/passwd auth with **login local**
- vty should only be config for SSH

**Enhancing Login:** Assigning passwds/local auth doesn't prevent being targeted
- Simply slows down attacks: Dictionary/DoS
- Detection profile enabling: Allows you to config device to react to repeated failed logins/blocking

**Quiet period:** A block can be config for period of time
**ACL's:** Can be used to perm legitimate connections from addresses of known sys
**Banners:** Disabled by default: Must be enabled: Protect org from legal perspective: Reviewed by legal counsel
- Never state anything like 'welcome' that could be misconstrued

**Config Login Enhancement**
**R1(config)# login block-for** *seconds* **attempts** *tries* **within** *seconds*
**R1(config)# login quiet-mode access-class [acl-name|acl-#]**
**R1(config)# login delay** *seconds*
**R1(config)# login on-success log [every** login**]**
Example:
**R1(config)# login block-for 15 attempts 5 within 60**
**R1(config)# ip access-list standard PERMIT-ADMIN**
**R1(config-std-nacl)# remark Permit only Administrative hosts**
**R1(config-std-nacl)# permit 192.168.10.10**
**R1(config-std-nacl)# permit 192.168.11.10**
**R1(config-std-nacl)# exit**
**R1(config)# login quiet-mode access-class PERMIT-ADMIN**
**R1(config)# login delay 10**
**R1(config)# login on-success log**
**R1(config)# login on-failure log**

**Enhancement cmds: Increase sec of virtual logins:**
**login block-for** Can defend against DoS attacks by disabling logins after # of failed attempts
**login quiet-mode** Maps to an ACL: ID's permitted hosts: Only auth hosts can attempt login
**login delay** Specifies # of seconds usr must wait bet unsuccessful attempts
**login on-success/failure** Logs successful/unsuccessful attempts

**THESE DO NOT APPLY TO CON connects:** Can only be enabled if local DB used for auth for local/remote access

- If lines config for passwd auth only: enhanced features not enabled

**Enable Login Enhancements**
**login block-for** Monitors login device activity
**Operates in 2 modes:**

| | |
|---|---|
| **Normal** | **AKA watch mode:** Rtr keeps count of # of failed attempts w/in amt of time |
| **Quiet** | **AKA quiet mode:** If # of failed logins exceeds config threshold: ALL attempts using Telnet/SSH/HTTP denied<br>• **ALL login attempts**: Including valid admin access: NOT perm<br>• **Override using ACL:** Created/ID'd using **quiet-mode access-class** |

**When implementing login block-for** 1s delay bet login attempts auto invoked

- Delay time bet attempts can be increased using **login delay** Occurs for all attempts: Failed/Successful

**Logging Failed Attempts**
**R1(config)# login on-success log** [every login]
**R1(config)# login on-failure log** [every login]
**R1(config)# security authentication failure rate** *threshold-rate* **log**

**R1# show login**
**R1# show login failures**

**3 cmds: Config to help detect passwd attacks:** Enables device to generate syslog msgs for failed/successful attempts
**login on-success/failure log** Generate syslog msgs for successful/unsuccessful logins

- # of attempts before logging msg generated cab be specified using [every login]
- **Default value is 1 attempt: Range is 1 – 65,535**

Alt to **login on-failure log | security authentication failure rate** Generates log msg when failure rate is exceeded
**show login** Verify **login block-for** settings/current mode
**show login failures** Displays addl. info regarding failed attempts (IP of attempts)

**Config SSH**
**R1(config)# ip domain-name span.com**
**R1(config)# crypto key generate rsa general-keys modulus 1024**
The name for the keys will be: R1.span.com
% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys, keys will be non-exportable…
[OK] (elapsed time was 2 seconds)
*JAN 17 21:21:21:41.977 %SSH-5-ENABLED: SSH 1.99 has been enabled

**R1(config)# ip ssh version 2**
**R1(config)# username Esther algorithm-type script secret cisco12345**
**R1(config)# line vty 0 4**
**R1(config-line)# login local**
**R1(config-line)# transport input ssh**
**R1(config-line)# end**

**R1# show crypto key mypubkey rsa**

**R1(config)# crypto key zeroize rsa**
% All keys will be removed.
% All router certs issues using these keys will also be removed.
Do you really want to remove these keys?

**4 requirements rtr must meet before config SSH:**

1. Cisco IOS release that supports SSH
2. Unique hostname
3. Contains correct domain name of network
4. Config for local auth/AAA services

**Steps:**

- **Config IP domain name of network using ip domain-name domain-name *[global]***
- 1-way secret keys must be generated so rtr can encrypt SSH traffic:
    - **Asymmetric**: IOS uses Rivest, Shamir, Adleman (RSA) alg to gen keys
    - **crypto key generate rsa general-keys modulus modulus-size** *[global]*
    - Modulus determines size of key/from 360-4,096 bits
    - Larger modulus: More secure key: Take longer to generate/encrypt/decrypt
    - Min key length: 1,024 bits
    - SSH auto enabled after RSA keys generated

- **Manually config ver 2 w/ip ssh version 2 *[global]***
- Ensure valid local DB usrname: No? Create one: username name algorithm-type scrypt secret secret
- Enable vty inbound/SSH sessions login local/transport input ssh

**Verify SSH/display generated keys**
**show crypto key mypubkey rsa** *[priv EXEC]*
**crypto key zeroize rsa** Overwrite/remove existing keys

**Mod SSH Config**
**R1# show ip ssh**

**R1(config)# ip ssh time-out 60**
**R1(config)# ip ssh authentication-retries 2**

**Verify optional SSH settings:**
**show ip ssh**
**ip ssh time-out seconds** *[global]* Mod default 120-second timeout: Config # of seconds SSH can use to auth usr

- After: EXEC session starts: standard exec-timeout config for vty applies

**Default:** Usr logging has 3 attempts to enter correct passwd before disconnect
**Config different # of SSH retries**
**ssh ip authentication-retries #** *[global]*

**Connecting to SSH-Enabled Rtr**
**show ssh** Verify status of client connections
**2 diff ways to connect:**
**Default: When SSH enabled: Cisco rtr can act as SSH server/client**
**As server:** Rtr can accept SSH client connects
**As client:** Rtr can connect via SSH to another SSH-enabled rtr

**Limiting Cmds:**
**Config Priv Lvl Syntax:** Config AAA/Issue show cmds/Config IDS/IPS/Config NetFlow
**WAN Engineer Privs:** Config routing/ints/issue show cmds
**16 Priv Lvls**

| | |
|---|---|
| **0** | Predefined for usr-lvl access privs. Seldom used: disable/enable/exit/help/logout |
| **1** | Default for login w/rtr prompt Router> Usr can't make any changes/view run config |
| **2-14** | Customized usr-lvl privs: Cmds from lower lvls may be moved up/down |
| **15** | Enable mode privs (enable cmd): Can change/view config files |

Syntax: **Router(config)# privilege mode** [**level** *level* | **reset**] **command**

| | |
|---|---|
| **mode** | Specifies config mode: Use privilege ? to see complete list avail on your rtr |
| **level** | Setting a priv lvl w/specified cmd (optional) |

| | |
|---|---|
| **level** | Priv lvl associated w/cmd: Can specify up to 16 lvls 0-15 (optional) |
| **reset** | Resets priv lvl of a cmd (optional) |
| **command** | Arg to use when you want to reset priv lvl (optional) |

**IOS: 2 methods of infrastructure access:**

1. Priv lvl
2. Role-based CLI: More granularity/control

**2 lvls of access to cmds:**

| | |
|---|---|
| **Usr EXEC (priv lvl 1)** | Lowest EXEC mode usr privs: Allows only usr-lvl cmds at router> prompt |
| **Priv EXEC (priv lvl 15)** | Includes all enable-lvl cmds at router# prompt |

**Config/Assign Priv Lvls**
**R1> enable 5**
Password: <cisco5>
**R1# show privilege**
Current privilege level is 5
**Syntax: privilege exec level level** Config priv lvl

| | |
|---|---|
| **lvl 5** | cmds avail for predefined lvl 1/ping cmd |
| **lvl 10** | cmds avail for lvl 5 && reload cmd |
| **lvl 15** | Predefined/doesn't need to be config: Access to all cmds including view/change config |

**2 methods for assigning passwds to diff priv lvls:**
**username name privilege level secret password** *[global]* Usr granted specific priv lvl: Assign to specific usr
**enable secret level level password** *[global]* To the priv lvl: Assign to specific EXEC mode passwd
**Both usrname secret/enable secret cmds config for type 9**

**Limitations:**

- No access control to specific ints/ports/logical ints/slots on rtr
- Assigning cmd w/multiple keywords allows access to all cmds that use those keywords

Example: Access to **show ip route** -> Allows access all **show**/**show ip** cmds
**Role-Based Views: CLI provides 3 views that dictate avail cmds:**

| | |
|---|---|
| **Root** | To config: Admin must be in root view: Same access priv as lvl 15: Not the same as lvl 15 usr<br>• Only root view usr can config new view/add/remove cmds from existing views |
| **CLI** | Specific set of cmds can be bundled: Unlike priv lvls: Has no cmd hierarchy/no higher/lower views<br>• Each view must be assigned all cmds associated with it<br>• Doesn't inherit cmds from any other view<br>• Same cmds can be used in multiple views |
| **Superview** | Consists of 1/more CLI views: Admins can define which cmds accepted/which config info visible<br>• Allows an admin to assign usrs/groups multiple CLI views at once<br>• Instead of single CLI view per usr w/all cmds associated w/it<br>**Superviews: Specific chars:**<br>• Single CLI view can be shared w/in multiple superviews<br>• Cmds can't be config for a superview: Admin must add cmds to CLI view & add that to superview<br>• Usrs logged into superview can access all cmds config'd for any CLI views part of that superview<br>• Each superview has a passwd used to switch bet superviews/from CLI view to superview<br>• Del a superview doesn't del associated CLI views: Remain avail to be assigned to another superview |

**Config Role-Based Views**
**Router# enable** [**view** [**view-name**]]

- Enters CLI view: Enter name root/specific view name: Root assumed if no name
- Must config **aaa new-model** cmd prior to entering

**Creates view/enters view config mode**
**Router(config)# parser view view-name**

**Sets a passwd to protect access to view**
**Router(config-view)# secret encrypted-password**

**Adds cmds/ints to a view**
**Router(config-view)# commands parser-mode** [**include | include-exclusive | exclude**] [**all**]

| | |
|---|---|
| **commands** | adds cmds/ints to view |
| **parser-mode** | mode in which specified cmd exists (Ex. EXEC mode) |
| **include** | adds cmd/int to view \| allows same cmd/int to be added to other views |
| **include-exclusive** | adds cmd/int to view \| excludes same cmd/int from being added to other views |
| **exclude** | excludes cmd/int from view |
| **all** | Wildcard: • Allows every cmd in specified config mode that begins w/same keyword • or every subint for a specified int to be part of view |
| **int int-name** | int added to view |
| **command** | cmd added to view |

```
R1(config)# aaa new-model
R1(config)# parser view SHOWVIEW

R1(config-view)# secret cisco
R1(config-view)# commands exec include show
R1(config-view)# exit

R1(config)# parser view REBOOTVIEW
R1(config-view)# secret cisco10
R1(config-view)# commands exec include reload
R1(config-view)# exit
```

**Before an admin can create a view:** AAA must be enabled **aaa new-model**
- To config/edit views: admin must log in as root view enable view privileged EXEC

**5 steps to create/manage specific view:**
1. Enable AAA **aaa new-model** *[global]*: Exit/enter root view w/**enable view**
2. Create view **parser view view-name** *[global]* View config mode: Max limit of 15 views total
3. Secret passwd to view **secret encrypted-password view** *[config]*
4. Assign cmds to selected view using **commands parser-mode** *[view config]*
5. Exit

**Config Role-Based CLI Superviews**
**R1(config)# parser view view-name superview**
- Appending keyword superview to parser view creates a superview/enters view config

**R1(config-view)# secret encrypted-password**
**R1(config-view)# view view-name**
- Adds CLI view to a superview
- Multiple views can be added
- Views may be shared bet superviews

```
R1(config)# parser view USER superview
R1(config-view)# secret cisco
R1(config-view)# view SHOWVIEW
R1(config-view)# exit
```

**Steps to config superview: Almost same as config CLI view:**
- **view view-name** Used to assign cmds to superview
- Admin must be in root view to config superview
- **enable view**/**enable view root** To check if you're in root view

**4 steps to create/manage superview:**
1. Create view parser **view view-name superview** *[superview config]*
2. Assign passwd **secret encrypted-password**

3. Assign existing view **view view-name** *[view config]*
4. Exit

**Verify Role-Based CLI Views: enable view**

- Enter name of view to verify/provide passwd to login: Use ? to verify cmds avail correct

**IOS Resilient Config**

- Faster recovery if something reformats flash mem/erases start config in NVRAM/oh shit moment
- Maintains working copy of IOS img file/copy of run config

**Primary bootset:** Refers to above: Files can't be rem by user

- No extra space required
- Auto detects img/config ver mismatch
- Only local storage used for securing files
- Can only be disabled through con session
- Only avail for sys that support PCMCIA flash int

**Enabling:**

```
R1(config)# secure boot-image
R1(config)# secure boot-config
R1(config)# exit
R1# show secure bootset
```

**secure boot-image** *[global]* Secure IOS img/enable img

- When enabled 1st time: Img secured/log entry generated

**secure boot-config** *[global]* Take snapshot of rtr run config/archive it in persistent storage

- Log msg displayed on con: Config archived/hidden

**show secure bootset** Verify existence of archive

**Primary Bootset Img**

```
Router# reload
<Issue Break sequence, if necessary>
rommon 1 > dir flash0:
rommon 2 > boot flash0:c1900-universal.mz.SPA.154-3.M2.bin
<Router reboots w/img>
R1(config)# secure boot-config restore flash0:rescue-cfg
R1(config)# end
Router# copy flash0:rescue-cfg running-config
```

**Restore primary bootset from secure archive:**

1. Reload rtr Issue the break sequence to enter ROMmon
2. Enter **dir** command to list contents of device that contains secure bootset file
3. Boot rtr w/secure bootset img boot: **flash mem loc: filename**
4. Enter global: Restore secure config to filename of choice **secure boot-config restore loc:file**
5. Exit

**Config Secure Copy**

```
R1(config)# ip domain-name span.com
R1(config)# crypto key generate rsa general-keys modulus 2048
R1(config)# username Esther privilege 15 algorithm-type script secret cisco12345
R1(config)# aaa new-model
R1(config)# aaa authentication login default local
R1(config)# aaa authorization exec default local
R1(config)# ip scp server enable
```

```
R2# copy flash0:R2backup.cfg.scp
R1# debug ip scp
```

**IOS Resilient:** Method for securing IOS img/config files locally on device: Uses SCP (Secure Copy Protocol)

**SCP remotely copies these files: Secure/auth method**

- Relies on SSH/requires AAA be config so rtr can determine whether usr has correct priv lvl

**Config rtr for server-side SCP w/local AAA:**

1. Config SSH
2. For local auth: Config at least 1 usr w/lvl 15

3. Enable **aaa new-model** [global]
4. Use **aaa authentication login default local** Specify local DB used for auth
5. Use **aaa authorization exec default local** Config cmd authorization
6. Enable SCP server-side functionality **ip scp server enable**

**Recovering Rtr Passwd**

1. Connect to con port
2. Record config register setting
3. Power cycle rtr
4. Issue break sequence
5. Change default config register w/ **confreg 0x2142** cmd
6. Reboot
7. Ctrl-C to skip setup procedure
8. Go into priv EXEC
9. Copy start config to run config
10. Verify config
11. Change enable secret passwd
12. Enable all ints
13. Return config-register to original setting from 2. Use **config-register** *[global]*
14. Save config files

**R1(config)# no service password-recovery**

- No args/keywords: All access to ROMmon disabled
- To recover: Initiate break sequence w/in 5s after img decompresses during boot
- If flash mem doesn't contain valid IOS img: ROMmon xmodem cmd can't be used to load new img
    - Admin would need to obtain a new img on a flash SIMM/PCMCIA card

# Post 3

Thursday, January 24, 2019 11:30 PM

SECURING NETWORK DEVICES P2

**Out-of-Band/In-Band Access**
When logging/managing info: Info flow bet hosts/devices: 2 paths

| | |
|---|---|
| **In-band** | Info flows across an enterprise/Internet/both: Regular chans<br>• Private encrypted tunnel/VPN tunnel<br>• In-band management occurs only when OOB management is not possible or available<br>• Apply only to devices: That need managed/monitored<br>• IPsec/SSH/SSL<br>• Should mgmt chan be open at all times?<br>• Smaller networks/lower cost sec deployment |
| **Out-of-band** | **OOB**: Info flows on dedicated mgmt network: No production traffic<br>• Mitigate risk of insecure protocols over production network<br>• Large enterprises |

**Mgmt devices**: Prevent comm w/other hosts on same subnet: Separate LAN segments/VLANs
**Intro to Syslog:** Notify admins w/msgs: Non-critical/significant: Most common method: Syslog: RFC 5424
**Syslog: UDP: Port 514:** Sends notifications across IP to collectors

- Many devices support [rtrs/switches/app servers/FW's/etc…]

**3 primary functions:**

1. Monitoring/troubleshooting
2. Selecting type of log info captured
3. Specify destinations of captured syslog msgs

**Syslog Op: Cisco:** Protocol: Sends sys msgs/debug output to local logging internally to device based on configs

- Rtrs can log: Config changes/ACL violations/int status/CPU utilization/etc…

**Set mem thresholds:**
**memory free low-watermark threshold io**
**memory free low-watermark processor**
**Can send msgs to diff facilities**

| | |
|---|---|
| **Logging buffer** | Msgs in rtr mem for short time: Cleared on reboot |
| **Console** | Default: **ON**: Sent to console where admin activates int |
| **Terminal lines** | EXEC sessions can be config to receive logs: Any term lines |
| **Syslog server** | Fwds log msgs to external syslog service |

**Syslog Message**

| Level | Keyword | Description | Definition |
|---|---|---|---|
| 0 | Emergencies | Sys unusable | LOG_EMERG |
| 1 | Alerts | Immed. action needed | LOG_ALERT |
| 2 | Critical | | LOG_CRIT |
| 3 | Errors | | LOG_ERR |
| 4 | Warnings | | LOG_WARNING |
| 5 | Notifications | Normal/significant | LOG_NOTICE |
| 6 | Info | | LOG_INFO |
| 7 | Debugging | | LOG_DEBUG |

| Syslog lvl/Name | Definition | Example |
|---|---|---|
| 0 LOG_EMERG | Panic condition normally broadcast to all usrs | IOS SW couldn't load |

| | | |
|---|---|---|
| 1 LOG_ALERT | Condition that should be corrected immediately | Temp too high |
| 2 LOG_CRIT | Critical event that needs attention | Unable to allocate mem |
| 3 LOG_ERR | Error occurred w/in device | Invalid mem size |
| 4 LOG_WARNING | Condition that may need to be evaluated | Crypto op failed |
| 5 LOG_NOTICE | Non-error may require special handling | Int state changed |
| 6 LOG_INFO | Normal event | Packet denied by ACL |
| 7 LOG_DEBUG | Msgs that contain info when debugging program | Packet type invalid |

**Cisco:** Produce syslog msgs from network events: Contains severity lvl/facility

- Smaller: MORE critical
- **Level 0-4**: SW/HW functionality
- **Level 5-6:** Notifications/info msgs
- **Level 7**: Various debug cmds

**Syslog facility:** Service identifiers that ID/categorize sys state data for error/event msg reporting: Options device specific

**000048: *Feb 01 11:22:33.779: %LINEPROTO–5–UPDOWN: Line protocol on int s0/0/0, changed state to up**

| | | |
|---|---|---|
| **000048** | Seq no | Stamps log msgs w/# if **service-sequence numbers**config |
| ***Feb 01 11:22:33** | Timestamp | If **service timestamps** config |
| **%LINEPROTO** | Facility | Source of cause of sys msg |
| **5** | Severity | Levels 0-7 |
| **UPDOWN** | MNEMONIC | Txt str uniquely describes msg |
| **Line protocol on..** | Description | Txt str: Detailed info about event |

**Syslog Sys: 2 types:**

| | |
|---|---|
| **Servers** | AKA: Log hosts: Accept/process log msgs from syslog clients |
| **Clients** | Rtrs/other equip that generate/fwd log msgs to syslog servers |

**Config Sys Logging**

**Router(config)# logging host** *[hostname* | *ip-address]*

- hostname: Specifies name of host to use as syslog server
- IP: Host you want to use as syslog server

**Router(config)# logging trap** *level* (0-7)
**Router(config)# logging source-interface** *int-type int-#*
**Router(config)# logging on**

**Steps:**

1. Set destination logging host: **logging host**
2. Log severity (trap) lvl: **logging trap**
3. Set source int: **logging source-interface**
   - Specifies that syslog packets contain IPv4/6 address of specific int
   - Regardless of which int packet uses to exit rtr
4. Enable logging to all enabled destinations: **logging on**
5. **show logging** View logging config/buffered syslog msgs

**Intro to SNMP: Simple Network Management Protocol**

- Developed to allow admins to manage devices on IP network
- Enables network monitoring (perf), managing devices, troubleshooting/growth

**Consists of 3 elements w/NMS: Network Mgmt Sys:**

1. SNMP manager
2. SNMP agents (managed node)
3. MIB: Mgmt Info Base

**SNMP managers/agents: UDP to exchange info**

- Agents listen: Port 161
- Managers: Listen: Port 162
- SNMP manager: SW: Can collect info from agent using get requests
- Can change configs on an agent using set requests
- Must be config to provide access to local MIB

- Can be config to fwd notifications (traps) directly to SNMP Manager

**MIB: Mgmt Info Base**

**SNMP set/get/trap msgs:** All have access to create/change info in MIB

- Info org hierarchically
- Each piece of info w/in MIB given OID: Object ID
    - Orgs OID's based on RFC standards into hierarchy
    - MIB tree includes branches w/vars common to devices/vendors
    - Cisco SNMP Object Navigator: Allows network admin to research details about an OID

**SNMP Vers**

| | |
|---|---|
| **SNMPv1** | RFC 1157: Provided no auth/encryption mech |
| **SNMPv2c** | RFCs 1901-1908: Improved on SNMPv1: Provided no auth/encryption mech |
| **SNMPv3** | RFCs 2273-2275: Secure access to devices by auth/encrypting packets over network |

| **Model** | **Level** | **Auth** | **Encryption** | **Result** |
|---|---|---|---|---|
| SNMPv1 | noAuthNoPriv | Comm str | No | Uses comm str match for auth |
| SNMPv2c | noAuthNoPriv | Comm str | No | Uses comm str match for auth |
| SNMPv3 | noAuthNoPriv | Username | No | Uses username match for auth |
| SNMPv3 | authNoPriv | MD5/SHA | No | Auth based on HMAC-MD5/HMAC-SHA algs |
| SNMPv3 | authPriv (reqs crypto SW img) | MD5/SHA | DES/AES | Auth based on HMAC-MD5/HMAC-SHA algs<br>Specify USM: User-Based Sec Model w/these encrypt algs<br>• DES 56-bit in add to auth based on CBC-DES standard<br>• 3DES 168-bit encryption<br>• AES 128/192/256-bit encryption |

**SNMP Vulns**

- At least 1 manager node should run SNMP mgmt SW
- Devices that can be managed [switches/rtrs/servers/workstations]: Equip w/SNMP agent SW module
- Agents responsible for providing SNMP manager access to local MIB: Stores data about device op

**Vuln**: Agents can be polled w/get requests/accept config changes w/set requests

**SNMPv3: 3 Sec features:**

| | |
|---|---|
| **Msg integrity/auth** | Ensures packet hasn't been tampered in transit: From a valid source |
| **Encryption** | Encrypts packet |
| **Access control** | Restricts each principal to certain actions on specific portions of data |

**Config SNMPv3 Sec**

**Config ACL to perm protected mgmt network:**

**Router(config)# ip access-list standard** *acl-name*

**Router(config-std-nacl)# permit source_net**

**Config SNMP view:**

**Router(config)# snmp-server view** *view-name* **oid-tree**

**Config SNMP group:**

**Router(config)# snmp-server group** *group-name* **v3 priv read** *view-name* **access** *[acl#* | *acl-name]*

**Config a usr as a member of the SNMP group:**

**Router(config)#**

**snmp-server user** *usrname* **group-name v3 auth** *[md5* | *sha]* **auth-passwd priv** *[des* | *3des* | *aes] [128* | *192* | *256] privpassword*

1. **Config ACL:** Permit access to auth SNMP managers
2. **Config SNMP view** w/**snmp-server view** ID MIB OIDS SNMP manager will read:

    - Config a view required to limit SNMP msgs to read-only access
3. **Config SNMP** group features w/**snmp-server group**
    - Config name for group
    - Set SNMP ver to 3 w/**v3** keyword
    - Require auth/encryption w/**priv** keyword
    - Associate view to group: Give it read only access w/**read**
    - Specify ACL config in Step 1
4. **Config SNMP group usr** features w/**snmp-server user**
    - Config usrname/associate usr w/group name from Step 3
    - Set SNMP ver to 3 w/**v3** keyword
    - Set auth type md5/sha/config auth passwd: SHA preferred
    - Require encryption w/**priv** keyword and config passwd

**Verify SNMPv3 Config**
**show snmp user** View usr info
- Use: SNMP mgmt tool: ManageEngine's SNMP MIB Browser: Config tool w/usr details
- When usr config: Use tool's features to test config usr can access SNMP agent
- Verify data encrypted by running Wireshark/capture SNMP packets

**NTP: Network Time Protocol**
- Ensure log msgs accurately time stamped by maintaining sync of clocks on hosts/devices

**Date/Time settings on rtr can be set w/2 methods:**
1. Manually edit the date and time
2. Config NTP
    - Allows rtrs on network to sync time settings w/NTP server
    - A group of NTP clients that obtain time/date info from single source have more consistent time settings
    - When NTP implemented: Can be set up to sync to private master clock/publicly avail NTP server
    - UDP: Port 123: RFC 1305

**NTP Server**
**If private master clock implemented:** Make sure valid source/sec site: Otherwise/vulns
**Vulns:** DoS via sending bogus NTP data across web to network in order to change clocks
- Can cause certs to become invalid **|** confuse admin **|** Mess w/order of syslog events on devices
- Many NTP servers: Don't require auth of peers

Comms (associations) bet machines running NTP: Statically config'd: Each device given IP of NTP masters

**ntp master** *[global]* 1/more rtrs get designated as master clock keeper (AKA NTP master)
- Clients: Either contact master/listen for msgs from master to sync clocks

**ntp server ip-address** Contact master
**ntp broadcast client** *[int config]* LAN: Use IP broadcast msgs
- Reduces config complexity: Each machine can be config to send/receive broadcast msgs
- Accuracy marginally reduced b/c info flow 1-way

**NTP Auth:** Ver3/Later: Supports crypto auth mech bet NTP peers: Can be used to help mitigate attack
**3 cmds used on NTP master/client:**
**ntp authenticate**
**ntp authentication-key key-number md5 key-value**
**ntp trusted-key key-number**
**Clients config w/out auth still get time from server:** Diff: Clients don't auth the server as sec source
**show ntp associations detail** Confirm server is auth source
**ntp server ntp-server-address** Set key-number value as an arg in cmd

**Discovery Protocols CDP/LLDP:** Cisco: Deploys many services enabled by default: To simplify config
- Some services can make devices vuln

| | |
|---|---|
| **CDP** | **Cisco Discovery Protocol**<br>• Enabled by default on Cisco rtrs |
| **LLDP** | **Link Layer Discovery Protocol**<br>• Open source standard can be enabled on Cisco devices/other vendors that support it<br>• Config/verification similar to CDP<br>**lldp run** *[global]*<br>**show cdp/lldp neighbors detail** Reveal device's address/platform/OS details |

**Attackers:** Don't need to have CDP/LLDP-enabled devices to gather sensitive info

- SW like Cisco CDP Monitor can be dl'd to gain info
- Intent CDP/LLDP? Make it easier for admins to discover/troubleshoot devices on network
    - Should be used w/caution: Shouldn't be everywhere: Especially not edge rtrs

**Impt practices avail to help ensure a device is sec:**
**Disable the following:**

- Unnecessary services/ints
- Commonly config mgmt services [SNMP]
- Probes/scans [ICMP]: Terminal access sec
- Gratuitous/proxy ARPs
- IP-directed broadcasts

**Cisco AutoSecure:** Released IOS 12.3: Initiated from CLI/exe's script

- Makes recommendations for fixing sec vulns/mods sec config of rtr
- Can lock down mgmt plane functions/fwding plane services/functions of rtr

**7 mgmt plane services/functions:**

1. Secure BOOTP/CDP/FTP/TFTP/PAD/UDP/TCP small servers/MOP/ICMP (redirects, mask-replies)/IP source routing/Finger/Passwd encryption/TCP keepalives/Gratuitous ARP/Proxy ARP/Directed broadcast
2. Legal notification using banner
3. Sec passwd/login functions
4. Sec NTP
5. Sec SSH
6. TCP intercept services

**3 fwding plane services/functions that AutoSecure enables:**

1. CEF: Cisco Express Forwarding: Traffic filtering w/ACLs
2. IOS FW inspection for common protocols
3. Often used in field to provide baseline sec policy on new rtr
    - Features can be altered after

**Using AutoSecure**
**auto secure** Enable AutoSecure setup: Can be interactive/non-interactive

| | |
|---|---|
| **Interactive mode** | Default: Rtr prompts options to enable/disable services/other sec features<br>**auto secure full** To config |
| **Non-interactive mode** | **auto secure no-interact**<br>Will auto exe AutoSec feature w/recommended default settings<br>Must be entered w/keywords to config specific components: Ex: management \| forwarding |

**When auto secure cmd initiated:** CLI wiz steps through config

1. auto secure Rtr displays config wizard welcome msg
2. Wiz gathers info about outside ints
3. Secs mgmt plane by disabling unnecessary services
4. Prompts for banner
5. Prompts for passwds/enables passwd/login
6. Ints secured
7. Fwding plane sec
8. When complete: Run config displays all config settings/changes

**Routing Protocol Spoofing:** Disrupting peer network rtrs/spoofing info carried w/in protocols

- Spoofing routing info generally used to cause systems to lie to each other/DoS/redirect traffic

**Consequences:**

- Redirecting traffic to create routing loops
- Redirecting traffic so it can be monitored on an insecure link
- Redirecting traffic to discard it

**OSPF MD5 R-Protocol Auth:** Can be enabled globally for all ints/per int
**Enable globally**:
**ip ospf message-digest-key key md5 password** *[int config]*
**area area-id authentication message-digest** *[rtr config]*

- Forces auth on all OSPF enabled ints
- If int not config w/ip ospf message-digest-key Will not be able to form adjacencies w/OSPF neighbors

**Enable MD5 auth on per int basis:**

**ip ospf message-digest-key key md5 password** *[int config]*
**ip ospf authentication message-digest** *[int config]*
- Int setting overrides global setting
- MD5 auth passwds don't have to be the same throughout an area
- Need to be same bet neighbors

**OSPF SHA R-Protocol Auth**: MD5 now considered vuln: Only should be used when no stronger auth avail
**2 major steps:**
1. Specify auth key chain [global]:
    - Config key chain name w/**key chain**
    - Assign key chain a #/passwd w/**key | key-string**
    - Specify SHA auth:  **cryptographic-algorithm**
    - Specify when key will expire w/**send-lifetime**
2. Assign auth key to desired ints w/**ip ospf authentication key-chain**

**Network Device Operations**
Primary function rtrs: Fwd usr-generated content across data plane: Generate/receive traffic destined for control/mgmt planes
- Rtrs must be able to distinguish bet data/control/mgmt plane packets to treat each one appropriately

| | |
|---|---|
| **Data plane packets** | Usr-generated packets:<br>• Always fwded by network devices to other end-station devices<br>• From perspective of network device:<br>• Data plane packets always have transit dest IP<br>• Can be handled by normal dest IP-based fwding processes |
| **Control plane packets** | Network device generated/received packets used for creation/op of network<br>Examples include:<br>• Protocols: OSPF/ARP/BGP/protocols that keep network converged/op properly<br>• Control plane packets: Generally sent to rtr/network device<br>• Dest IP is of rtr |
| **Mgmt plane packets** | Network device generated/received packets used to manage network<br>Examples include:<br>• Protocols: Telnet/SSH/SNMP/NTP/others used to manage device/network |

**Normal op:** Vast majority of packets handled by network devices: Data planes
- Handled by CEF: Cisco Express Forwarding

**Cisco Express Forwarding:** Uses control plane to pre-populate CEF FIB: Fwding Info Base table in data plane
- W/appropriate egress int for given packet flow
- Subsequent packets that flow bet same source/dest. fwded by data plane based on info contained in FIB

**Control/Mgmt Plane Vulns**
- Rtr processor (CPU in Control Plane): Significantly less capable of handling kinds of packet rates exp by CEF
- Never directly involved in fwding of data plane packets

**In contrast:** When high packet rates overload control/mgmt plane, route processor resources can be overwhelmed
- Reduces avail of these resources for tasks critical to op/maintenance of network
- Malicious/non-malicious events can overwhelm route processor resources
    - Events can include: Crafted packet attacks/High rates of packets directed at control plane
    - Non-malicious events? Rtr/network misconfigs/SW bugs/network failure re-convergence events
    - Important to take appropriate steps to protect route processor from being overwhelmed

**Int ACL:** Traditional/most avail approach for managing packets entering/exiting network device
- ACLs well understood/generally applicable to data/services/control/mgmt plane packets
- Applied at int lvl to each packet ingressing (or egressing) the int: Not just control plane packets
- ACLs must be applied to every int to which policy to be applied

**CoPP: Control Plane Policing Operation**
IOS: Designed to allow admins manage flow of traffic that is "punted" to route processor
- Punt: Defined by Cisco: Action an int takes when sending a packet to route processor

- CoPP: Designed to prevent unnecessary traffic from overwhelming route processor

Protects route processor on network devices by treating route processor resources as separate entity w/own int

- A CoPP policy can be dev/applied to only those packets w/in the control pane
- Unlike int ACLs: No effort wasted investigating data plane packets that will never reach control plane

# Post 4

Thursday, January 24, 2019 11:30 PM

IMPLEMENTING FW TECHS P1

**AAA: Auth/Authorization/Accounting:** Framework for scalability/access sec
Cisco: AAA access local usr/pass db: Better sec: Cost effective: Easy implement
Large org: AAA auth against ACS: Cisco Secure Access System: Scalable: All devices: Access central server

- Fault tolerant: Multiple servers

**Cisco ISE: Cisco Identify Services Engine:** Visibility into usrs/devices accessing |Enforce policies for endpoint devices

- Simple mgmt: Device profiling/posture assessment/guest mgmt/ID-based access

**802.1X**: Access LAN sec: Port-based access control/auth protocol

- Restricts unauth workstations from connecting to LAN through publicly accessible switch ports

**Auth w/out AAA**
**Syntax:**

```
R1(config)# line vty 0 4
R1(config-line)# password cisco
R1(config-line)# login

R1(config)# ip domain-name esther-moo.com
R1(config)# crypto key generate rsa general-keys modulus 2048
R1(config)# username Admin algorithm-type script secret password
R1(config)# line vty 0 4
R1(config-line)# transport input ssh
R1(config-line)# login local
```

**Access controls:** Limits who/what can use specific resources/services/options avail after access

- Simple method of remote access auth: Config login/passwd combo on con/vty/aux lines/ports
- Least sec: No accountability

**SSH: More sec remote access:** Req both usrname/passwd: Encrypted during transmission:

- Local DB method provides addl sec: Attacker req to know usrname/passwd
- Accountability: User recorded when login

**Local DB limitations**: Usr accts must be config locally on each device: Not for large networks/No fallback auth method
**Better to:** Have devices refer to same DB of usrnames/passwds from central server
**AAA Components**
**Authenticate:** Who permit access
**Authorize**: What can do while there
**Accounting**: Auditing actions performed while access
**3 Func Components:**

| | |
|---|---|
| **Auth** | Usrs/Admins: Must prove they are who they say they are:<br>• Established? Usr/pass combos, challenge/response questions, token cards, etc… |
| **Authorization** | After auth: Determines which resources usr accesses/ops can be perform |
| **Accting/Audit** | Records what usr does: What accessed/amt of time/any changes<br>• Tracks how resources used |

**AAA Auth Modes:** Used auth usrs for admin/remote access
**2 common methods:**

| | |
|---|---|
| **Local Auth** | Uses local DB for auth: AKA self-contained auth/local AAA auth<br>• Stores usrnames/passwds locally in rtr<br>• Usrs auth against local DB<br>• DB same 1 req for establishing role-based CLI<br>• Small networks |
| **Server-Based Auth** | Rtr accesses central server: ACS for Win |

| | |
|---|---|
| | • Contains usr/pass for all usrs<br>• Rtr uses RADIUS/TACACS+ protocols to comm w/AAA server<br>○ **RADIUS: Remote Auth Dial-In User Service**<br>○ **TACACS+: Terminal Access Controller Access Control System**<br>• For multiple rtrs/switches |

**Authorization:** What usrs can/can't do on network after auth: Similar to priv lvls/role-based CLI rights/privs

- Typically uses server-based solution
- Created set of attributes describes usr's access
- Compared to info contained in DB: Restrictions for usr made
- Auto/doesn't req usrs perform addl steps after auth: Implemented after auth

**Accounting:** Collects reports/data: Auditing/billing:

- Start/stop connection times/exe cmds/# of packets/bytes
- Implemented using server-based: Service reports back to ACS server
- Stats can be extracted to create detailed reports about config
- Combine w/AAA auth: Helps manage access to internetworking devices: More sec
- Logs everything

**Types of Accounting info:**

| | |
|---|---|
| **Network** | Captures info: All PPP protocol sessions: Packets/byte counts |
| **Connection** | Captures info: All outbound connections from AAA: Example: Telnet/SSH |
| **EXEC** | Captures info: About EXEC term sessions (usr shells) on access server: usrname/date/start-stop times/IP |
| **System** | Captures info: All sys-lvl events: Example: When sys reboots/when accting turned on/off |
| **Cmd** | Captures info: EXEC shell cmds for specific priv lvl being exe on access server<br>• Each cmd record includes: List of cmds exe for that priv lvl/date-time exe/usr who exe |
| **Resource** | Cisco: Captures start-stop record support for calls that have passed usr auth:<br>• Stop records calls that fail to auth as part of usr auth<br>• Necessary for usrs employing accting records to manage/monitor networks |

**Auth Admin Access**

**Syntax:**

```
R1(config)# username JR-ADMIN algorithm-type script secret cisco
R1(config)# username ADMIN algorithm-type script secret cisco1
R1(config)# aaa new-model
R1(config)# aaa authentication login default local-case
```

- Local AAA auth: Smaller networks: Local usrnames/passwds stored on rtr

Admin must fill local sec DB by specifying usr/pass for EACH

**Config local AAA services to auth admin access:**

1. Add usr/pass to local router DB that need admin access to rtr
2. Enable AAA globally on rtr
3. Config AAA params on rtr
4. Confirm/troubleshoot configs

**aaa authentication login** Allows ADMIN/JR-ADMIN usrs to log into rtr via con/vty term lines
**default** Auth applies to all lines: Except spec line config overrides default
**local-case** Auth case sensitive: Both usr/pass

**Auth Methods:**

```
R1(config-line)# aaa authentication login [ default | list-name] method 1 [method 2 method 3]
```

| Cmd | Description |
|---|---|
| **default** | Uses listed auth methods that follow this keyword as default list of methods when usr logs in |
| **list-name** | Char str used to name list of auth methods activated when usr logs in |
| **method 1…** | ID's list of methods AAA auth process will query in a given sequence: At least 1 must be specified<br>• Max 4 methods may be specified |

**Method Type Keywords**

| | |
|---|---|
| **enable** | enable passwd for auth |
| **local** | local usrname DB for auth |
| **local-case** | case-sensitive local usrname auth |

| | |
|---|---|
| **none** | No auth |
| **group radius** | List of all RADIUS servers for auth |
| **group tacacs+** | List of all TACACS+ servers for auth |
| **group group-name** | Subset of RADIUS/TACACS+ servers for auth<br>**aaa group server radius**<br>**aaa group server tacacs+** |

**aaa new-model** *[global]* Enable AAA
**no aaa new-model** *[global]* Disable AAA

- When **aaa new-model** used: Unseen default auth using local DB auto applied to all lines except con
- Always config local DB entry before enabling AAA

**aaa auth login** Enable auth of con/aux/vty lines
**default** Applies auth to all lines

- Custom auth method can be config using list-name

**Methods:** Cmd ID's type of methods queried to auth usrs:

- Up to 4 methods can be defined: Fallback if one method doesn't work
- When usr attempts to login: 1st method listed used
- SW attempts auth w/next listed auth
- If auth method denies usr access: Stops and no other methods allowed

**local | local-case** Enable local auth using preconfig local DB

- **local** Accepts usrname regardless of case
- **local-case** Case sensitive

**enable** Specify usr can auth using enable password

- Ensure auth succeeds even if all methods return error: Specify none as final method
- Use none when testing

**Default/Named Methods:**
**aaa authentication login list-name** Apply diff method lists to diff ints/lines
**no authentication login** Possible to return to default method list
**Syntax:**

```
R1(config)# username JR-ADMIN algorithm-type script secret cisco
R1(config)# username ADMIN algorithm-type script secret cisco
R1(config)# aaa new-model
R1(config)# aaa authentication login default local-case enable
R1(config)# aaa authentication login SSH-LOGIN local-case
R1(config)# line vty 0 4
R1(config-line)# login authentication SSH-LOGIN
```

**Debug Options:** Useful for troubleshooting auth issues:
**debug aaa** 7 keywords can be used: Interpreted by control plane: Load on rtr resources: Impact performance
**Debugging AAA Auth**
**debug aaa authentication** Look for **GETUSER/GETPASS** msgs: Helpful when ID'ing method lists referenced

- **PASS** msg: Successful login

**no debug aaa authentication** Disable debugging for aaa auth
**undebug all** Disable debugging
**ACS: Cisco Secure Access Control System:** Centralized: Ties enterprise's access policy/ID strategy: Scalable: High perf

- Can be leveraged to control admin access/config for all devices supporting RADIUS/TACACS+/both

**TACACS+/RADIUS: Both auth protocols used to comm w/AAA servers:** Each supports diff capabilities/func

| | |
|---|---|
| **3 Factors TACACS+** | 1. Separates auth/authorization\\<br>2. Encrypts all comm<br>3. TCP port 49 |
| **4 Factors RADIUS** | 1. Combines RADIUS auth/authorization as 1 process<br>2. Encrypts only passwd<br>3. UDP<br>4. Supports remote-access tech: 802.1X/SIP: Session Initiation Protocol |

**TACACS+:** More secure: Protocol exchanges encrypted
**RADIUS:** Only encrypts usrs passwd: Doesn't encrypt usrnames/accting info/other info carried in msgs

| | TACACS+ | RADIUS |
|---|---|---|
| **Functionality** | Separates AAA according to arch: Allows modularity | Combines auth/auth but separates accting Less flexibility than TACACS+ |
| **Standard** | Mostly Cisco | Open/RFC Standard |
| **Transport Protocol** | TCP | UDP |
| **CHAP** | Bidirectional challenge/response | Unidirectional challenge/response from server to client |
| **Confidentiality** | Entire packet encrypted | Passwd encrypted |
| **Customization** | Authorization of rtr cmds per-usr/per-group basis | No option to authorize rtr cmds No per-usr/per-group basis |
| **Accting** | Limited | Extensive |

**TACACS+ Auth:** Cisco enhancement to original TACACS: Entirely new: Incompatible w/any previous vers

- Separate AAA services: Flexibility: Possible to use for authorization/accting/another method
- Extensions provide more types of auth reqs/response codes than original TACACS specification
- Multiprotocol support: IP/AppleTalk
- Encrypts entire body of packet for more sec comms
- TCP port 49

**RADIUS Auth:** Dev by Livingston Enterprises: Open IETF standard AAA protocol for apps

- Both local/roaming: Commonly used for accting purposes
- RFCs: 2865/2866/2867/2868/3162/6911
- Hides passwds during trans: EVEN w/PAP: Passwd Auth Protocol
  - Using op that involves MD5 hashing/shared secret
  - Rest of packet: Plaintext

**Combines auth/authorization as 1 process:**

- When usr auth: Usr also authorized
- UDP port 1645/1812 for auth
- UDP port 1646/1813 for accting

**Widely used by VoIP SP's:**

- Passes login creds of SIP endpoint (broadband phone) to SIP registrar using digest auth
- Then to a RADIUS server: Using RADIUS
- Common auth protocol utilized by 802.1X

**DIAMETER AAA Protocol:** Next gen protocol alternative to RADIUS:

- IETF standard
- Uses new transport protocol: SCTP: Stream Control Trans Protocol
- TCP instead of UDP

Integration of TACACS+/ACS

**Cisco Secure ACS 5.6 Features:**

- Distributed arch for medium/large-scale deployments
- Lightweight web-GUI w/intuitive navigation: Both IPv4/IPv6 clients
- Admin auth through MS AD and LDAP
- Automated reports sent through email
- Integrated advanced monitoring/reporting/troubleshooting: Excellent control/visibility using SNMP traps
- Encrypted syslogs
- Full auditing/reporting capabilities

**Integration of AAA/AD:**

**AD domain controller:** Used to enforce sec policies by auth/authorizing usrs when they log into Win domain

- Can also be used to handle auth/authorization on IOS devices
- ACS can be integrated to use AD service: MS Win Server can be config as AAA server
- MS implementation of AAA server using RADIUS is known as IAS: Internet Auth Service
- Starting w/Server 2008: IAS renamed: NPS: Network Policy Server

**Config for IOS same as comm w/any RADIUS server: Only diff:**

- MS AD controller used to perform auth/authorization services

**Integration w/ISE: Identity Service Engine:**

**Cisco ISE: Identity/Access control policy platform:** Enables enterprise to:

- Enforce compliance/enhance infrastructure sec/streamline service ops
- Arch of ISE: Gathers real-time contextual info from networks/usrs/devices
- Admin can use info: Make proactive gov decisions to tying ID to various elements
    - Can include: Access switches/WLAN/WLCs: Wireless LAN Controllers/VPNs/gateways/Data center switches
- BYOD: ISE defines access policies/enforces compliances

**TrustSec:** Main policy component: Protests data/apps/mobile from unauth access

- Combines policy def/control/reporting in 1 appliance

**4 features of ISE toolset:**

| | |
|---|---|
| **Device profiling** | Determine whether personal/corporate device |
| **Posture assessment** | Determine if device clean of viruses/suspicious apps before entering network<br>• Can also make sure device's AV SW up to date |
| **Guest mgmt** | Grants/enforces temp access for guests |
| **AAA** | Auth/Authorization/Accting into 1 app w/device profiling/posture assess/guest mgmt |

Primary function: ID based network access:

**Context-aware ID mgmt:**

- Determine whether usrs accessing network on authorized/policy-compliant device
- Establish usr ID/loc/access history: Compliance/reporting
- Assign services based on assigned usr role/group/associated policy [job role/loc/device type/etc…]
- Grant auth usrs access to specific network segments: Specific apps/services: Both based on auth results

**Config Server-Based AAA Auth:**

1. Globally enable AAA to allow use of all AAA elements: Pre-req for all other AAA cmds
2. Specify Cisco Secure ACS that will provide AAA for rtr: TACACS+/RADIUS
3. Config encryption key needed to encrypt data transfer bet network access server/ACS
4. Config AAA auth method list to refer to TACACS+/RADIUS: Redundancy: Possible to config more than 1 server

**Syntax:**

```
R1(config)# aaa new-model
R1(config)# tacacs server Server-T
R1(config-server-tacacs)# address ipv4 192.168.1.101
R1(config-server-tacacs)# single-connection
R1(config-server-tacacs)# key TACACS-class
R1(config-server-tacacs)# exit
```

**aaa new-model** Enable AAA
**tacacs server** name Config the server name
**address IPv4** Address of TACACS+ server: Option to modify auth/accting port
**single-connection** Enhance TCP performance by maintaining single TCP connections for life of session

- Otherwise: TCP connection opened/closed otherwise: Default: TCP connection open/close for each session
- TACACS+: Req multiple servers can be ID by entering addresses

key Used to config shared key to encrypt data transfer bet TACACS+ sever/AAA-enabled rtr

- Must be config exactly same on both rtr/TACACS+ server

**Configuring RADIUS Servers**
**Syntax:**

```
R1(config)# aaa new-model
R1(config)# radius server SERVER-R
R1(config-radius-server)# address ipv4 192.168.1.100 auth-port 1812 acct-port 1813
R1(config-radius-server)# key RADIUS-class
R1(config-radius-server)# exit
```

**radius server name** Config RADIUS server: Puts you into server config mode

- UDP: No equiv single-connection keyword
- If req: Multiple RADIUS servers can be ID by entering radius server name cmd for each server

**address ipv4 IP** Config RADIUS server address

- Default: Cisco rtrs use port 1645 for auth and 1646 for accting
- IANA: Reserved ports 1812 for RADIUS auth port: 1813 RADIUS accting port
- Impt to make sure ports match between rtr/RADIUS server

**key** Config shared key for encrypting the password

- Key must be config same on rtr/RADIUS server

**Config Auth to use AAA Server**
**Syntax:**

```
R1(config)# aaa new-model
R1(config)# tacacs server server-T
R1(config-server-tacacs)# address ipv4 192.168.1.100
R1(config-server-tacacs)# key TACACS-class
R1(config-server-tacacs)# exit

R1(config)# radius server SERVER-R
R1(config-radius-server)# address ipv4 192.168.1.101 auth-port 1812 acct-port 1813
R1(config-radius-server)# key RADIUS-class
R1(config-radius-server)# exit
```

**When AAA sec servers ID'd:** Servers must include method list of aaa authentication login
**group tacacs+/radius**: ID's AAA servers

**Monitor Auth Traffic**

```
R1# debug aaa authentication
AAA Authentication debugging is on
14:01:17: AAA/AUTHEN (567936829): Method-TACACS+
14:01:17: TAC+: send AUTHEN/CONT packet
14:01:17: TAC+: (567936829): received authen response status = PASS
14:01:17: AAA/AUTHEN (567936829): status = PASS
```

**debug aaa authentication** High-lvl view of login activity

**Debugging TACACS+/RADIUS**
**debug radius**
**debug tacacs**
**debug tacacs events** *[priv EXEC*]

- Displays open/close TCP connections/bytes read/written over connection/TCP status
- Can generate lots of output

**Config Cisco Rtr to Access AAA RADIUS Server**

1. Create usrs on RADIUS server
2. Set secret key on RADIUS server
3. Verify port 1812 for RADIUS auth-port/1813 for RADIUS accting-port
4. Set up SSH on rtr
5. Set up local usr on rtr in case of server fail
6. Enable AAA auth on rtr
7. Set AAA auth login method lists
8. Enable rtr to use RADIUS server for auth by config on the router:
    - Server name
    - Server IP/auth port 1812/acct-port 1813
    - Shared secret key
9. Config con line/specify AAA login auth method list to use
10. Config VTY lines for SSH/specify AAA login auth method list to use
11. Test/verify

**AAA Autho Config**
**Syntax:**
**R1(config)# aaa authorization [ network | exec | commands level ] [default | list-name ] method1…[method4]**
**Syntax:**

```
R1(config)# username JR-ADMIN algorithm-type script secret class
R1(config)# username ADMIN algorithm-type script secret class
R1(config)# aaa new-model
R1(config)# aaa authorization exec default group tacacs+
R1(config)# aaa authorization network default group tacacs+
```

**aaa authorization** Config authorization
**Specify types of cmds/services:**

| | |
|---|---|
| **network** | Network services such as PPP |
| **exec** | Starting exec (shell) |
| **cmds lvl** | Exec (shell) cmds |

When authorization not enabled: ALL usrs allowed full access: After auth started: Default: Allow no access
**AAA Accting Config**
**R1(config)#  aaa accounting [ network | exec | connection ] [ default | list-name ] [ start-stop | stop-only | none ] [broadcast] method1…[method4]**

**Syntax:**

```
R1(config)# username JR-ADMIN algorithm-type script secret class
R1(config)# username ADMIN algorithm-type scrypt secret class
R1(config)# aaa new-model
R1(config)# aaa authentication login default group tacacs+
R1(config)# aaa authorization exec default group tacacs+
R1(config)# aaa authorization network default group tacacs+
R1(config)# aaa accounting exec default start-stop group tacacs+
R1(config)# aaa accounting network default start-stop group tacacs+
```

**aaa accounting** Config AAA accting
**Commonly used aaa accting keywords:**

| | |
|---|---|
| **network** | Runs accting for all network-related service requests (incl. PPP) |
| **exec** | Accting for EXEC shell session |
| **connection** | Accting on all outbound connections (SSH/Telnet) |

**Next:** Record type/trigger config: Trigger specifies what actions cause accting records to be updated
**Possible triggers:**

| | |
|---|---|
| **start-stop** | Sends start/stop accting notice at beginning/end of a process |
| **stop-only** | Sends stop accting record for all cases including auth failures |
| **none** | Disables accting services on line/int |

# Post 5

Thursday, January 24, 2019 11:31 PM

802.1X QUICKIE: CH.3 LAST

**802.1X Port-Based Auth:** 802.1X standard defines port-based access control/auth protocol

- Restricts workstations from connecting to a LAN through publicly accessible switch ports
- Auth server taps each workstation connected to a switch port before making any services offered avail

**Devices in network have specific roles:**

| | |
|---|---|
| **Supplicant (Client)** | Device that reqs access to LAN/switch services: Responds to reqs from switch<br>• Must be running 802.1X-compliant SW<br>• Port client attached to is supplicant [client] |
| **Authenticat(Switch)** | Controls phys access to network based on auth status of client<br>• Switch acts as intermediary (proxy) bet client (supplicant)/auth server<br>• Req ID'ing info from client: Verifying that info w/auth server/relaying response to client<br>• Switch uses RADIUS SW agent<br>○ Responsible for encapsulating/de-encapsulating EAP: Extensible Auth Protocol frames<br>○ Also w/interacting w/auth server |
| **Auth server** | Actual auth of client: Auth server validates ID of client/notifies switch<br>• Whether client autho to access LAN/switch services<br>• B/C switch acts as proxy: Auth service transparent to client<br>• RADIUS sec sys w/EAP ext only supported auth server |

**Until workstation auth: 802.1X access control enables only EAPOL: EAP over LAN**

- Through port to which workstation connected
- After auth succeeds: Normal traffic can pass through it
- Switch port state determines whether client is granted access to network

**When config for 802.1X port-based auth:**

**Port starts in unauth state:**

- While in state: Port disallows all ingress/egress traffic except 802.1X packets
- When client successfully auth: Port transitions to autho state: Allows all traffic for client to flow

If switch reqs client ID (authenticator initiation)/client doesn't support 802.1X

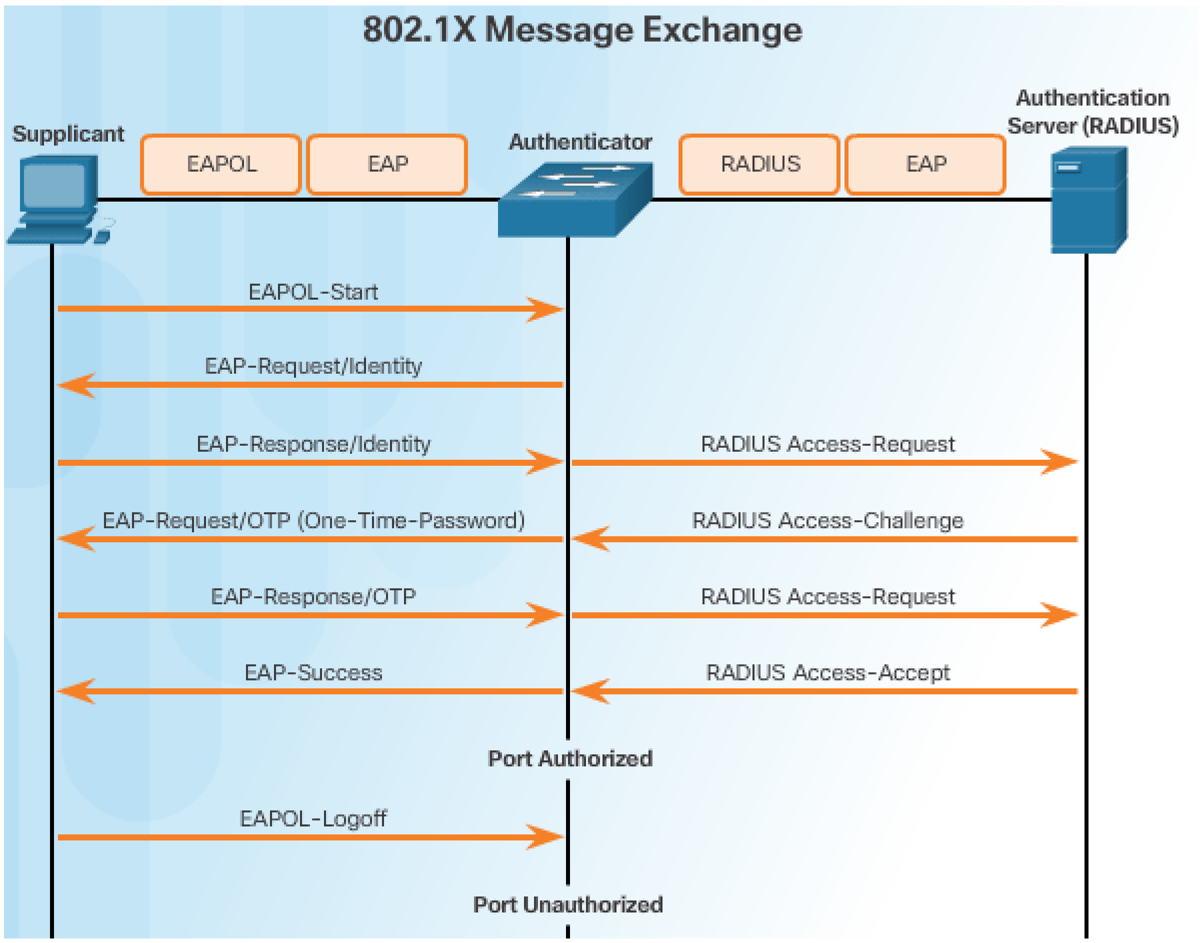
- Port remains unauth: Client not granted access

**When 802.1X-enabled client connects to port/client initiates auth process (supplicant initiation):**

- By sending EAPOL-start frame to switch not running 802.1X:
  - No response received
  - Client begins sending frames as if port in auth state

**Encapsulation as follows:**

| | |
|---|---|
| **Bet supplicant/authenticator** | EAP data encapsulated in EAPOL frames |
| **Bet authenticator/auth server** | EAP data encapsulated using RADIUS |



**802.1X Port Autho State**
**S1(config-if)# authentication port-control [ auto | force-authorized | force-unauthrorized ]**

| Param | Description |
|---|---|
| **auto** | 802.1X port-based auth enable: Causes port to begin in unauthorized state<br>• Enables only EAPOL frames to be sent/received through port |
| **force-authorized** | Default: Port sends/received normal traffic w/out 802.1X based auth of client |
| **force-unauthorized** | Port remains in unauthorized state: Ignores all attempts by client to auth<br>• Switch can't provide auth services to client through port |

If client auth's: Port state changes to auth: All frames from auth client enabled through port
If auth fails: Port remains in unauth stats: Auth can be retried
**Client logs out?** Sends EAPOL-logout msg: Causes switch port to trans to unauth state

**authentication port-control** Control the port auth state
**auto** Must be used to enable 802.1X auth

**Config**
```
S1(config)#  aaa new-model
S1(config)# radius server CCNAS
S1(config-radius-server)# address ipv4 10.1.1.50 auth-port 1812 acct-port 1813
S1(config-radius-server)# key RADIUS-class
S1(config-radius-server)# exit
S1(config)# aaa authentication dot1x default group radius
S1(config)# dot1x system-auth-control

S1(config)# int fa0/1
S1(config-if)# description Access Port
S1(config-if)# switchport mode access
S1(config-if)# authentication port-control auto
S1(config-if)# dot1x pae authenticator
```

**Config 802.1X:**

1. **aaa new-model** Enable AAA/config RADIUS server
2. **aaa authentication dot1x** Create 802.1X port-based auth method list
3. **dot1x system-auth-control** Globally enable 802.1X port-based auth
4. **authentication port-control** Enable port-based auth on int
5. **dot1x pae** Enable 802.1X auth on int: Authenticator options sets PAE: Port Access Entity type
    - Int acts only as authenticator/won't respond to any msgs meant for supplicant

# Post 6

Thursday, January 24, 2019 11:31 PM

CH4. IMPLEMENTING FW TECH

**ACLs: Access Control Lists:** Widely used in sec for mitigating attacks/controlling traffic: Can be define for: L2/3/4/7
**Historically: Type of ACL could be ID'd by #**

- **Range: 200–299:** Control traffic according to Eth type
- **Range: 700–799:** Traffic is classified/controlled based on MACs

**Today:** Most common uses IPv4/6 addr and TCP/UDP ports

- **Standard/extended** IPv4 ACLs can be named/numbered **| IPv6 ACLs** must use name

**Config Numbered/Named ACLs**
**ACL:** Sequential list of perm/deny statements: Known as ACE's: Access Control Entries
**ACE:** Also called ACL statements: **Filter traffic on criteria:**

- Source/dest addr
- Protocol
- Port #'s

**Instead of using #:** Name can be used to config: Must be specified as standard/extended
**access-list {acl-#} { permit | deny | remark }** *source-addr* **[***source-wildcard***] [log]**

| Param | Description |
|---|---|
| **acl-#** | Dec # from 1-99 or 1300-1999 |
| **deny** | Denies access if conditions matched |
| **permit** | Permits access if conditions matched |
| **remark** | Add remark about entries in IP ACL to make list easier to understand |
| **source-addr** | # of network/host from which packet being sent<br>**2 ways to specify:**<br>1. 32-bit quantity in 4-part dotted dec fmt<br>2. Keyword any as abbreviation for source/wildcard of 0.0.0.0 255.255.255.255 |
| **source-wildcard** | Optional: 32-bit wildcard mask to be applied to source: Places 1's in bit positions you want to ignore |
| **log** | Optional: Causes an info logging msg about packet that matches entry to be sent to con<br>• Lvl of msg logged is controlled by logging console<br>**Msg includes:**<br>• ACL #<br>• Whether packet perm/deny<br>• Source addr<br>• # of packets<br>• Generated for 1st packet that matches:<br>○ Then at 5min intervals/including # of packets perm/deny in prior 5 min interval |

**Extended Numbered ACL's**
**access-list {acl-#} { permit | deny | remark }** *protocol source-addr* **[*source wildcard*]** *dest-addr***[***dest-wildcard***] [op port] [established]**

| Param | Description |
|---|---|
| **acl-#** | ID's ACL using # range: 100-199 (extended) or 2000-2699 (expanded) |
| **protocol** | Common keywords: **icmp/ip/tcp/udp**<br>• To match any protocol [incl ICMP/TCP/UDP] use ip keyword |
| **dest-addr** | # of network/host which packet is being sent |
| **dest-wildcard** | To be applied to dest |
| **operator** | Optional: Compares source/destination ports<br>Operands include:<br>• **lt** (less than) |

| | |
|---|---|
| | • **gt** (greater than)<br>• **eq** (equal)<br>• **neq** (not equal)<br>• **range** (inclusive range) |
| **port** | Optional: Dec # of a TCP/UDP port |
| **established** | Optional: TCP only: Established connection |

**Name ACL: R1(config)# ip access-list [ standard | extended ] name_here**
**Config ACEs: Access Control Entries**
**Standard ACL:** Match packets by examining source IP in header of that packet

- Used to filter packets solely on L3 source info

**R1(config-std-nacl)# [ permit | deny | remark ]** *source* **[***source-wildcard***] | any**
**Extended ACL:** Match packets based on L3/L4 source/dest info | L4 can include: TCP/UDP port info

- Greater flexibility/control over standard ACLs

**R1(config-std-nacl)# [ permit | deny | remark ] protocol** *source-addr* **[***source-wildcard***]** *dest-addr* **[***dest-wildcard***] [operator port]**
**Applying:** After creating: Can apply diff ways
**Apply ACL to int**
**R1(config-if)# ip access-group {acl-#|name} in | out**
**Apply ACL to VTY**
**R1(config-line)# access-class {acl-#|name} in | out**

**show access-list** See how many packets have matched a statement
**log** Affects performance: Only use when network under attack

**ACL Config Guidelines:** ACL made up of 1/more ACEs/statements

- Create ACL globally: Apply it
- Ensure last statement is implicit **deny any/deny any any**
- Remember statement order is impt: Processed top-down: As soon as statement matched: ACL is exited
- Ensure most specific statements are at top of list
- **Only 1 ACL allowed per int/per protocol/per direction**
- New statements for existing ACL added to bottom of ACL
- Rtr generated packets not filtered by outbound ACLs

**Standard ACLs: As close to dest as possible**
**Extended ACLs: As close to source as possible**
**Editing ACLs:** Default: Sequence #ing happens in increments of 10

- Assigned to each ACE w/in ACL
- After ACL created/applied: Can be edited using these seq #'s
- Use seq #'s to del/add specific ACEs at various places
- If # not specified for new entry: Rtr auto places it at bottom of list/assigns appropriate #

**Example:** 3 entries
**R1# show access-lists**
Extended IP access list 101
10 permit tcp any any
20 permit udp any any
30 permit icmp any any

**Editing/Adding new ACE: Replacing ACE in line 20**
**R1(config)# ip access-list extended 101**
**R1(config-ext-nacl)# no 20**
**R1(config-ext-nacl)# 5 deny tcp any any eq telnet**
**R1(config-ext-nacl)# 20 deny udp any any**

**R1# show access-lists**
Extended IP access list 101
5 tcp any any eq telnet
10 permit tcp any any
20 deny udp any any
30 permit icmp any any

**Seq #'s/Standard ACLs:** Standard: IOS applies internal logic to config ACEs/verify ACLs

- Host statements (specific IPv4 addr) listed 1st
- Not necessarily in order entered
- IOS puts host statements in particular order determined by hashing function

**Antispoofing w/ACLs:** Can be used to mitigate many threats: IP spoofing/DoS:

- Most DoS attacks use some spoofing
- IP spoofing overrides normal packet creation by inserting custom IP header w/diff source -addr
- Many well-known classes of IP's should never be seen as source addr for traffic entering org

**Perm Necessary Traffic through FW: Effective strategy for mitigating attacks:**

- Explicitly permit only certain types of traffic through
- Example: DNS/SMTP/FTP are services that often must be allowed
- Common to config FW so it permits admins remote access through FW
- SSH/syslog/SNMP examples of services rtr may need to include
- Should be controlled/monitored

**Mitigating ICMP Abuse**

- ICMP echo packets [pings] can be used to discover subnets/hosts/generate DoS floods
- Can use ICMP redirect msgs to alter host r-tables
- ICMP echo/redirect msgs should be blocked inbound by rtr

**Should be allowed into internal:**

| | |
|---|---|
| **Echo reply** | Allows usrs to ping external hosts |
| **Source quench** | Req sender decrease traffic rate of msgs |
| **Unreachable** | Generated for packets denied by ACL |

**Req for proper op/should be allowed to exit:**

| | |
|---|---|
| **Echo** | Allows usrs to ping external hosts |
| **Param problem** | Informs host of packet header problems |
| **Packet too big** | Enables packet MTU: Max Trans Unit discovery |
| **Source quench** | Throttles down traffic when necessary |

**Rule:** Block all other ICMP msg types outbound
**Mitigating SNMP Exploits**
**R1(config)# no snmp-server**
Mgmt protocols like SNMP: Useful for remote monitoring/mgmt of networked devices

- Can still be exploited

**If SNMP necessary:**

- Exploitation of vulns can be mitigated by applying int ACLs to filter SNMP packets from non-auth sys
- Exploit may still be possible if SNMP packet sourced from spoofed addr/perm by ACL
- Most effective: Disable SNMP server on devices not required

**IPv6 ACLs: IPv4 designed w/out modern-day req:**

| | |
|---|---|
| **Sec** | IPSec |
| **Device roaming** | Mobile |
| **QoS** | Resource Reservation Protocol: RSVP |
| **Addr scalability** | DHCP/NAT/CIDR/VLSM |

As migration continues: Attacks more pervasive
Example:
**Dual stack: Attackers leveraging IPv4 to exploit IPv6**

- **Dual stack:** Integration method which device has implementation/connectivity to both IPv4/6 networks
- Result: Device has 2 protocol stacks

**Attackers can accomplish stealth attacks that result in trust exploitation by using**

- Dual stacked hosts/rogue NDP: Neighbor Discovery Protocol (NDP) msgs/tunneling techniques

**Teredo tunneling**: IPv6 transition tech: Provides auto v6 addr assignment when IPv4/IPv6 hosts located behind v4 NAT devices

- Embeds v6 packets inside v4 UDP packets
- Compromised host sends rogue router ads: Triggers dual stacked hosts to obtain IPv6 addr
- Can then use foothold to move around/pivot inside network
- Can compromise addl hosts before sending traffic back out network

**IPv6 ACL**
**R1(config)# ipv6 access-list name-here**
**R1(config-ipv6-acl)# deny | permit *protocol* [***source-ipv6-prefix/length* **| any|** *host source-ipv6-addr***]**
**[operator port#] [***dest-ipv6-prefix/length* **| any |** *host dest-ipv6-addr***] [operator port#]**

| Param | Description |
|---|---|
| **deny\|permit** | Deny/perm packet |
| **protocol** | Name/# of protocol/integer representing ipv6 protocol # |
| **source-dest** | Source/dest ipv6 network/class of to set conditions |
| **any** | any as an abbrev for prefix ::/0<br>• Matches all addr |
| **host** | Source/dest ipv6 to set conditions |
| **operator** | Optional: Compares source/dest ports of specified protocol<br>• **lt** (less than)<br>• **gt** (greater than)<br>• **eq** (equal)<br>• **neq** (not equal)<br>• **range** |
| **port#** | Optional: Dec # name of TCP/UDP port for filtering |

**Similar to IPv4 but:**
- No equiv to IPv4 standard ACLs
- All IPv6 ACLs must be config w/name.
- Allow filtering based on source/dest addr traveling in/out to int
- Support traffic filtering based on IPv6 option headers/upper-layer protocol type info: Similar to extended ACL's in IPv4

**ipv6 traffic-filter** Apply ACL to int
**IPv6 ACL contains**
- implicit deny ipv6 any
- implicit permit rules to enable IPv6 neighbor discovery

**IPv6 NDP requires use of network layer to send:**
1. **NA: Neighbor advertisements**
2. **NS: Neighbor solicitations**

If admin config deny ipv6 any w/out perm neighbor discovery: NDP will be disabled

**Defining Firewalls:** Originally: Fireproof wall: Stone/metal prevented flames from spreading to connected structures
**Later:** Applied to metal sheet that separated engine from passenger compartments of vehicles/aircraft
**Eventually:** Adapted for use w/networks: Prevents undesirable traffic from entering areas w/in them
**Common properties:**
- Resistant to attacks
- Only transit point bet networks b/c all traffic flows through them
- Enforce access control policy

| | |
|---|---|
| **1988** | **DEC: Digital Equip Corp created 1st FW in packet filtering form**<br>• Inspected packets to see if they matched sets of rules: Then chose to fwd/drop them accordingly.<br>• **AKA Stateless filtering:** Occurs regardless of whether packet part of existing flow of data<br>• Each packet filtered based solely on values of certain params in header: Similar to how ACLs filter packet |
| **1989** | **AT&T Bell Labs dev 1st stateful FW**<br>**Stateful FW:** Evaluates state of connections in data flows to filter packets<br>• Able to determine if packet belongs to existing flow of data<br>• Static rules are supplemented w/dynamic ones/created in real time to define active flows<br>• Help mitigate DoS attacks that exploit active connections through devices |

**Original FW's**: Not standalone devices: Rtrs/servers w/SW features added to provide funct
- Over time, companies dev standalones
- Dedicated FW's enabled rtrs/switches to off-load mem/processor-intensive activity of filtering packets

- Modern rtrs: ISR: Integrated Services Rtrs can also be used as sophisticated stateful FW's

**Firewall:** Sys/group of sys that enforce access control policy bet networks

| | |
|---|---|
| **Benefits** | Prevent exposure of sensitive hosts/resources/apps to untrusted usrs<br>• Sanitize protocol flow: Prevents exploitation of protocol flaws<br>• Block malicious data from servers/clients<br>• Reduce sec mgmt complexity by off-loading most of network access control to FW's in network |
| **Limitations** | Misconfig can have consequences: Single point of failure<br>• Data from many apps can't be passed over<br>• Network performance can slow<br>• Unauthorized traffic can be tunneled/hidden as legitimate traffic |

**Type Descriptions**

| | |
|---|---|
| **Packet filtering** | Typically rtr w/capability to filter some packet content, such as L3/ L4 info |
| **Stateful** | Monitors state of connections, whether it's in an initiation/data transfer/termination state: L3/4/5 |
| **App gateway (proxy)** | Filters info at L3/4/5/7: Most of the control/filtering done in SW<br>• When client needs to access remote server: Connects to proxy server<br>• Proxy connects to remote server on behalf of client: Server only sees connection from proxy |

**Other methods of implementing FWs:**

| | |
|---|---|
| **Host-based (serve/personal)** | PC/server w/FW SW on it |
| **Transparent** | Filters IP traffic bet pair of bridged ints |
| **Hybrid** | Combo of various FW types<br>Example: App inspection FW combines stateful FW w/app gateway FW |

**Packet Filtering FW/Limitations**

- Usually part of rtr FW: Permits/deny traffic based on L3/L4 info
- Stateless: Use simple policy table look-up that filters traffic based on criteria

Example: SMTP servers listen on 25: Default:

- Admin can config packet filtering to block 25 from specific machine to prevent from broadcasting email virus

**Advantages/Disadvantages: Packet Filtering FW:**

| | |
|---|---|
| **Advantages** | Don't represent complete FW solution: Impt element of FW sec policy:<br>• Implement simple permit/deny rules<br>• Low impact on network performance<br>• Easy to implement/supported by most rtrs<br>• Degree of sec at network layer<br>• Perform almost all tasks of high-end FW: Much lower cost |
| **Disadvantages** | Susceptible to spoofing: Can send arbitrary packets that meet ACL/pass through filter<br>• Don't reliably filter fragmented packets<br>○ Fragmented packets carry TCP header in 1st fragment/packet filter on TCP header info<br>○ All fragments after 1st are passed unconditionally<br>○ Decisions to use packet filters assume filter of 1st fragment accurately enforces policy<br>• Use complex ACLs: Diff to implement/maintain<br>• Can't dynamically filter certain services<br>Example: Sessions that use dynamic negotiations<br>• Diff to filter w/out opening access to range of ports<br>• Stateless: Examine each packet individually rather than context of state of connection |

**Stateful FWs:**

Most versatile/common in use: Provide stateful packet filtering by using connection info maintained in state table

- Stateful filtering: Arch at network layer: Analyzes traffic at L4/L5
- Unlike stateless that uses static packet filtering: Stateful filtering tracks each connection traversing

all ints of FW
    - Confirms they are valid
- Use state table to keep track of actual comm process
- Examines info in headers of L3 packets/L4 segments
    - Example: FW looks at TCP header for SYN/RST/ACK/FIN/other control codes to determine state of connection

**Each time TCP/UDP connection established:** Inbound/outbound: FW logs info in state table for that specific flow
- Note: Previous ver of IOS FW implemented stateful behavior: Newer: Zone-based

**Stateful FW: Benefits/Limitations**

| | |
|---|---|
| **Benefits** | Often used as primary means of defense by filtering unwanted/unnecessary/undesirable traffic<br>• Strengthen packet filtering by providing more stringent control over sec<br>• Improve performance over packet filters/proxy servers<br>• Defend against spoofing/DoS by determining whether packets belong to existing connection/unauth source<br>• More log info than packet filtering FW |
| **Limitations** | Can't prevent App Layer attacks b/c they don't examine actual contents of HTTP connection<br>• Not all protocols stateful<br>Example: UDP/ICMP don't generate connection info for state table<br>• Don't garner much support for filtering<br>• Diff to track connections that use dynamic port negotiation<br>• Some apps open multiple connections<br>• Reqs whole new range of ports that must be opened to allow 2nd connection<br>• Don't support usr auth |

**Next Gen FW's: Go beyond stateful in impt ways:**
1. Granular ID/visibility/control of behaviors w/in apps
2. Restricting web/web app use based on rep of site
3. Proactive protection against threats
4. Enforcement of policies based on usr/device/role/app type/threat profile
5. NAT/VPN/SPI: Stateful Protocol Inspection
6. Integrated IPS

**2014: Cisco: I**ntegration of Sourcefire's FirePOWER services into Cisco **ASA: Adaptive Security Appliance**
- Advanced malware protection
- Called ASA Next-Gen FW b/c adaptive/threat-focused
- Designed to provide defense across entire attack continuum: Before/during/after attacks

**Classic FW: Cisco IOS Classic: Formerly CBAC: Context-Based Access Control**
- Stateful FW feature added to IOS prior to 12

**4 main functions:**
1. Traffic filtering
2. Traffic inspection
3. Intrusion detection
4. Generation of audits/alerts

**Can examine supported connections for embedded NAT/PAT:** Port Addr Translation/perform necessary addr translations
- Can block P2P connections/instant msging traffic
- Only provides filtering for protocols specified by admin
- If protocol not specified, existing ACLs determine how it's filtered/no temp opening created
- Only detects/protects against attacks that travel through FW
- Doesn't typically protect against attacks w/in protected network unless it travels through internal rtr w/IOS FW enabled

**Classic Op:** Creates temp openings in ACL to allow returning traffic
Entries created as inspected traffic leaves network/rem when connection terms/idle timeout reached
1. **When traffic 1st generated:** Passes through rtr: Inbound ACL processed
    - If ACL denies connection type: Packet dropped
    - If ACL permits connection: Classic FW inspection rules examined
2. **Based on inspection rules:** IOS might inspect connection: If traffic not inspected/packet is allowed: No other info gathered
3. **If not:** Connection info compared to entries in state table: If connection doesn't exist: Entry added

    - If it exists: Idle timer for connection reset
4. **If new entry added:** Dynamic ACL entry added to allow returning traffic part of same connection
    - Temp opening only active for as long as session open: Entries NOT saved to NVRAM
5. **When session ends:** Dynamic info from state table/dynamic ACL entry rem

**Classic FW:** Can be config to inspect traffic in 2 directions: in/out
- Useful when protecting 2 parts of network: Both sides initiate certain connections/allow returning traffic to reach source

| | |
|---|---|
| **Classic FW Config** | 1. Choose internal/external ints<br>2. Config ACLs for each int<br>3. Define inspection rules<br>4. Apply inspection rule to in |

**Denying traffic based on source/dest/type**
**FW w/2 ints config:**
- Traffic originating from private permitted/inspected as it travels toward public
- Inspected traffic returning from public/associated w/traffic originated from private permitted
- Traffic originating from public/traveling to private generally blocked

**DMZ: Demilitarized Zones:**
A FW design where there is 1 inside int connected to private: 1 outside int connected to public work: 1 DMZ int
- Traffic originating from private inspected as travels toward public/DMZ
    - Permitted w/little/no restriction
    - Inspected traffic returning from DMZ/public to private permitted
- Traffic originating from DMZ/traveling to private usually blocked
- Traffic originating from DMZ/traveling to public selectively permitted based on service reqs
- Traffic originating from public/traveling toward DMZ selectively permitted/inspected
    - Email/DNS/HTTP/HTTPS traffic: Return traffic from DMZ to public dynamically permitted
- Traffic originating from public/traveling to private blocked

| **ZPFs** | Concept of zones to provide addl flexibility |
|---|---|
| **Zone** | Group of 1/more ints that have similar functions/features<br>• Help specify where IOS FW should be applied<br>• Default: Traffic bet ints in same zone not subject to any policy/passes freely<br>• All zone-to-zone traffic blocked<br>• In order to permit traffic bet zones: Policy allowing/inspecting traffic must be config |

**Only exception to default:** Rtr self zone:
**Self zone** is rtr itself/includes all rtr int IP's
- Policy configs that include self zone would apply to traffic destined to/sourced from rtr
- Default: No policy for type of traffic

**Traffic that should be considered when designing policy for self zone:**
- Mgmt plane
- Control plane [SSH/SNMP/r- protocols]

**Layered Defense**

| | |
|---|---|
| **Network Core** | Protects against malicious SW/traffic anomalies<br>• Enforces network policies<br>• Ensures survivability |
| **Perimeter** | Secs boundaries bet zones |
| **Endpoint** | Provides ID/device sec policy compliance |
| **Comm** | Info assurance |

**Layered defense uses diff types of FW's combined in layers to add depth to sec of org**
- Policies can be enforced bet layers/inside them
- Policy enforcement points determine whether traffic fwded/discarded

Example: Traffic that comes in from untrusted network 1st encounters packet filter on edge rtr
- If allowed by policy: Traffic goes to the screened FW/bastion host sys that applies more rules to traffic/discards suspect packets

**Bastion host:** A hardened computer typically located in DMZ
- Then traffic goes to interior screening rtr:

- Moves to internal dest host only after passing through all policy enforcement points bet outside rtr/inside network
- Type of DMZ setup called: Screened subnet config

**Admin must consider many factors when building complete in-depth defense:**

| | |
|---|---|
| **FW's don't** | ○ Stop intrusions that come from hosts w/in network/zone<br>○ Don't protect against rogue AP installs<br>○ Don't replace backup/disaster recov mechs resulting from attack/HW failure<br>○ Not a subs for informed admins/usrs |
| **Best Practice** | ○ Position FW's at sec boundaries<br>○ Critical part of sec: Unwise to rely exclusively on them for sec<br>○ Deny all traffic by default: Permit only needed services<br>○ Ensure phys access is controlled<br>○ Regularly monitor logs<br>○ Practice change mgmt for config changes |

**Benefits of ZPF**

**2 config models for IOS FW:**

| | |
|---|---|
| **Classic** | Traditional config model: FW policy applied on ints |
| **ZPF** | New config mode: Ints assigned to sec zones: FW policy applied to traffic moving bet them |

**Benefits of a ZPF:**
- Not dependent on ACLs
- Rtr sec posture is to block unless explicitly allowed
- Policies easy to read/troubleshoot w/C3PL: Cisco Common Classification Policy Lang
  - Structured method to create traffic policies based on events/conditions/actions
  - Scalable: Having 1 policy affect given traffic, instead of multiple ACLs/inspection actions

**ZPF Design: Common designs are LAN-to-Internet**

**Designing ZPFs involves:**
1. **Determining zones:** Admin focuses on separation of network into zones
2. **Establish policies bet zones:** For each pair of "source-dest" zones:
   - Define sessions clients in source zones can req from servers in dest zones
   - Most often TCP/UDP sessions: May be ICMP: Echo
   - For traffic not based on concept of sessions: Admin must define unidirectional traffic flows from source to dest
3. **Design phys infrastructure:** After zones ID'd/Traffic reqs bet doc: Design phys infrastructure
   - Must take into acct sec/avail reqs when designing phys infrastructure
   - Includes dictating # of devices bet most-sec/least-sec zones/determining redundant devices
4. **ID subsets w/in zones/merge traffic reqs:** Each FW device in design: Admin must ID zone subsets connected to ints
   - Merge traffic reqs for those zones

**ZPF Actions: 3 possible:**

| | |
|---|---|
| **Inspect** | IOS stateful packet inspection |
| **Drop** | Analogous to deny statement: Log option avail to log rejected packets |
| **Pass** | Analogous to permit statement: Doesn't track state of connections/sessions w/in traffic |

**Rules for Transit Traffic**: Traffic transiting through rtr ints subject to rules governing int behavior
- If neither int is zone member: Action to pass traffic
- If both ints members of same zone: Action is to pass traffic
- If 1 int is zone member [other's aren't]: Drop traffic regardless of whether zone-pair exists
- If both ints belong to same zone-pair/policy exists: Inspect/allow/drop as defined

**Rules for Traffic to Self Zone:**
- Self zone is rtr itself: Includes all IP's assigned to rtr ints

**Rules for ZPF are diff for self zone:**
- Rules depend on whether rtr is source/dest of traffic
- If rtr is: All traffic perm
- Only exception is if source/dest are zone-pair w/specific service-policy

| Source Int Member of Zone? | Dest Int Member of Zone? | Zone-Pair Exists? | Policy Exists? | Result |
|---|---|---|---|---|
| Yes (self zone) | Yes | No | N/A | PASS |

| | | | | |
|---|---|---|---|---|
| Yes (self zone) | Yes | Yes | No | PASS |
| Yes (self zone) | Yes | Yes | Yes | INSPECT |
| Yes | Yes (self zone) | No | N/A | PASS |
| Yes | Yes (self zone) | Yes | No | PASS |
| Yes | Yes (self zone) | Yes | Yes | INSPECT |

**Config ZPF:** Configs must be completed in order

1. Create the zones
2. ID traffic w/class-map
3. Define an action w/policy-map
4. ID zone pair/match it to policy-map
5. Assign zones to appropriate ints

**Create Zones**

- What ints should be included in zones?
- Name for each zone?
- What traffic necessary bet zones/which direction?

```
R1(config)# zone security zone-Name
```

**Example:**

```
R1(config)# zone security PRIVATE
R1(config-sec-zone)# exit
R1(config)# zone security PUBLIC
```

**ID Traffic**

```
R1(config)# class-map type inspect [match-any| match-all] class-map-name
```

| Param | Description |
|---|---|
| **match-any** | Packets must meet one of the match criteria to be considered a member of the class |
| **match-all** | Packets must meet all of the match criteria to be considered a member of the class |
| **class-map-name** | Name of class-map used to config policy for class in policy-map |

**Class-map sub-config Syntax**

```
R1(config-cmap)# match access-group {acl-# | acl-name }
R1(config-cmap)# match protocol protocol-name
R1(config-cmap)# match class-map class-map name
```

| Param | Description |
|---|---|
| **match access-group** | Config match criteria for a class-map based on the specified ACL #/name |
| **match protocol** | Config the match criteria for a class-map based on specified protocol |
| **match class-map** | Uses another class-map to ID traffic |

```
R1(config)# class-map type inspect match-any HTTP-TRAFFIC
R1(config-cmap)# match protocol http
R1(config-cmap)# match protocol https
R1(config-cmap)# match protocol dns
```

**Define an Action:** Use a policy-map to define what action should be taken for traffic that is a member of a class

**Action:** A specific functionality: Associated w/traffic class

| | |
|---|---|
| **Inspect** | Action offers state-based traffic control<br>• Example: If traffic traveling from PRIVATE zone to PUBLIC is inspected: Rtr maintains connection/session info for TCP/UDP<br>• Rtr would then permit return traffic sent from PUBLIC zone hosts in reply to PRIVATE zone connection reqs |
| **Drop** | Default: Similar to implicit deny any at end of ACL: Explicit drop applied by IOS to end of every policy—map<br>• Listed as class class-default in last section of any policy-map config<br>• Other class—maps w/in policy—map can also be config to drop unwanted traffic<br>• Unlike ACLs: Traffic silently dropped/no ICMP unreachable msgs sent to source of traffic |
| **Pass** | Action allows rtr to fwd traffic from 1 zone to another<br>• Pass action doesn't track state of connections |

| | |
|---|---|
| | • Pass only allows traffic in 1 direction<br>• A corresponding policy must be applied to allow return traffic to pass in the opposite direction<br>• Ideal for sec protocols w/predictable behavior, such as IPsec<br>• Most app traffic better handled in ZPF w/inspect action |

**R1(config)# policy-map type inspect policy-map-name**
**R1(config-pmap)# class type inspect class-map-name**
**R1(config-pmap-c)# { inspect | drop | pass }**
**Example:**
**R1(config)# policy-map type inspect PRIV-TO-PUB-POLICY**
**R1(config-pmap)# class type inspect HTTP-TRAFFIC**
**R1(config-pmap-c)# inspect**
**ID Zone-Pair/Match to Policy**
**R1(config)# zone-pair security zone-pair-name source { source-zone-name | self } destination {destination-zone-name | self }**
**R1(config-sec-zone-pair)# service-policy type inspect policy-map-name**

| Param | Description |
|---|---|
| **source source-zone-name** | Name of zone where traffic is originating |
| **destination destination-zone-name** | Name of zone traffic is destined |
| **self** | Specifies system-defined zone: Indicates whether traffic will be going to/from rtr itself |

**Example:**
**R1(config)# zone-pair security PRIV-PUB source PRIVATE destination PUBLIC**
**R1(config-sec-zone-pair)# service-policy type inspect PRIV-TO-PUB-POLICY**
**Assign Zones to Ints**
**R1(config-if)# zone-member security zone-name**
**Example:**
**R1(config)# int g0/0**
**R1(config-if)# zone-member security PRIVATE**
**R1(config-if)# int s0/0/0**
**R1(config-if)# zone-member security PUBLIC**
**Verify ZPF Config**
**R1# sh run | begin class-map**
**R1# sh policy-map type inspect zone-pair sessions**
**R1# show class-map type inspect**
**R1# show zone security**
**R1# show zone-pair security**
**R1# show policy-map type inspect**

**ZPF Config Considerations**
**Factors to consider:**
1. Rtr never filters traffic bet ints in same zone
2. An int can't belong to multiple zones: To create union of sec zones: Specify new zone/policy map/zone pairs
3. ZPF can coexist w/Classic although they can't be used on same int
   ○ Remove ip inspect int config cmd before applying zone-member sec cmd
4. Traffic can never flow bet int assigned to zone/int w/out zone assignment:
   ○ Applying zone-member config cmd always results in temp interruption of service until other zone-member config
5. Default inter-zone policy is to drop all traffic unless specifically allowed by service-policy config for zone-pair
6. zone-member cmd does not protect rtr itself (traffic to/from rtr isn't affected)
   ○ Unless zone-pairs are config using predefined self zone

**Other:**
- No filtering applied for intra-zone traffic
- Only 1 zone allowed per int
- No Classic FW/Zone-Based Policy FW config on same int
- If only 1 zone member assigned: All traffic dropped
- Only explicitly allowed traffic is fwded bet zones.

- Traffic to self zone not filtered.

**Create ACLs to define which traffic can go from zone to zone:**

- Create zones
- Assign zones to ints
- Create class maps to ID traffic
- Create policy maps to drop/inspect/pass traffic
- Assign class map(s) to policy map
- Config zone pairs
- Associate appropriate policy map to zone pair
- Test/verify

**Implementing FW Tech:** FW's separate protected areas from non-protected areas

- Prevents unauth usrs from accessing protected resources

**2 common methods:**

| | |
|---|---|
| **Packet Filtering FW** | Rtr w/capability to filter packet content: L3/L4 info using ACLs |
| **Stateful FW** | Monitors state of connections: Whether in an initiation/data transfer/termination state |

Standard/Extended IP ACLs can be used to provide packet filtering FW capabilities

- They are fundamental tools used for basic traffic filtering/mitigate wide range of attacks

**Stateful FW's implemented 3 ways**:

| | |
|---|---|
| **Traffic filtering** | Includes:<br>• ACLs using TCP established option/reflexive ACLs that extend func to consider 2-way nature of traffic |
| **IOS Classic FW** | Formerly CBAC: Stateful filtering of most forms of modern app traffic: Relies on ACLs/inspection rules applied on ints |
| **ZPF** | 2006: Config centers on creation of zones associated w/various areas of network/meant for diff lvls of sec<br>• Implement is structured/easier to understand than CBAC<br>• ZPF uses class-maps/policy maps to classify/filter traffic |

# Post 7

Thursday, January 24, 2019 11:32 PM

RTR/FW SEC: CH 7: CRYPTOGRAPHIC SYSTEMS

**3 objectives of secured comm:**

1. **Authentication:** Msg is real/comes from who it says it does
2. **Integrity:** No one intercepted/altered msg/like checksum function in frame
3. **Confidentiality:** If msg captured: Can't be deciphered

**Authentication:** 2 methods for validating source in network comm:

| | |
|---|---|
| Authentication services | Data nonrepudiation services |

| | |
|---|---|
| **Authentication** | Guarantees msg comes from source it claims it did |
| **Nonrepudiation** | Allows sender of msg to be uniquely ID: Sender can't deny being source of msg |
| **Integrity** | Ensures msgs not altered in transit: Receiver can verify msg identical: No manipulation |
| **Confidentiality** | Ensures privacy so only receiver can read msg: Encryption |
| **Encryption** | Process of scrambling data so it can't be easily read by unauth parties |
| **Decryption** | Reversed process: Key req to encrypt/decrypt msg |

**History: Various algs/methods used:**
**Caesar cipher:** Julius msgd by putting 2 sets of alphabet side-by-side: Shifted by specific #
**Hash function:** Transforms str of chars into fixed-length value/key that represents original str

- Diff bet hashing/encryption: How data stored | Encrypted text: Decrypted w/key
- Hashing: After data converted using hash: Plaintext gone

**Over centuries:** Various cipher methods/phys devices/aids used to encrypt/decrypt txt:

| | | | |
|---|---|---|---|
| Scytale | Caesar Cipher | Vigenère Cipher | Enigma Machine |

**Ciphertext:**

| | | |
|---|---|---|
| Transposition | Substitution | One-time pad |

**Transposition Ciphers:** No letters replaced: Rearranged
Example: FLANK EAST ATTACK AT DAWN
NWAD TAKCATTA TSAE KNALF
**Rail fence cipher:** Transposition where words are spelled out as if they were a rail fence

- Modern encryption algs [DES/3DES] still use transposition as part of alg

**Substitution Cipher:** Substitute 1 letter for another: Retain the letter frequency of original msg

- **Caesar cipher**: Msg relied on single shift: **Monoalphabetic** substitution
- **Polyalphabetic** cipher: Vigenère: Giovan Battista Bellaso: 1553
    - Later misattributed to French diplomat: Blaise de Vigenère

**OTP Ciphers:** Gilbert Vernam: AT&T Bell: 1917: Invented/patented stream cipher: Co-invented 1-time pad cipher

- Vernam proposed teletype cipher: Prepared key consisting of long/non-repeating seq of #'s on tape
- Combined char by char w/plaintxt msg to produce ciphertxt

**To decipher:** Same paper tape key combined char by char producing plaintext

- Each tape only used 1x: One-time pad

**Difficulties:** Creating random data: PC's incapable of creating true random data: If used more than 1x: Easy to break
**Cracking Code**:
**Cryptanalysis:** Study of determining meaning of encrypted info, w/out access to shared secret key
**History:**

- Vigenère cipher: Broken in 19th century by English crypto Charles Babbage
- Mary, Queen of Scots: Plotted to overthrow Queen Elizabeth I from throne
- Sent encrypted msgs to co-conspirators
- Cracking code used in plot led to beheading Mary: 1587

**Enigma-encrypted comm used by Germans to navigate/direct U-boats in Atlantic**

- Polish/British cryptanalysts broke German Enigma code
- Winston Churchill believed it was turning point: WWII

**Methods used in cryptanalysis:**

| | |
|---|---|
| **Brute-force** | Trying every possible key knowing eventually 1 will work |
| **Ciphertxt** | Attacker has ciphertxt of encrypted msgs but no knowledge of underlying plaintext |
| **Known-Plaintext** | Access to ciphertext of msgs/knows something about plaintext underlying ciphertext |
| **Chosen-Plaintext** | Choosing which data encryption device encrypts/observe ciphertext output |
| **Chosen-Ciphertext** | Choosing diff ciphertext to be decrypted/has access to decrypted plaintext |
| **Meet-in-the-Middle** | Attacker knows portion of plaintext/corresponding ciphertext |

**Cryptology:** Science of making/breaking secret codes
**2 separate disciplines:**

1. **Crypto:** Dev/use of codes
2. **Cryptanalysis:** Breaking of codes

**Crypto Hash Function:** Hashes: Used for integrity assurance

- Takes bin data: msg: Produces fixed-length/condensed representation: Hash
- AKA: Msg digest/digest/digital fingerprint
- Based on 1-way math func relatively easy to compute: Harder to reverse

**Applied in diff situations:** Proof of authenticity when used w/symmetric secret auth key [IPSec]: R-protocol auth

- Auth generating 1-time/1-way responses to challenges in auth protocols [CHAP]
- Msg integrity check proof: Digitally signed contracts: PKI: Public Key Infrastructure certs
    - Like those accepted when accessing secure site using browser

**Properties:** Hash func H takes an input x: Returns fixed-size str called hash value h: **h= H(x)**
**Should have following properties:**

- **Input:** Any length **||** Output: Fixed length
- H(x): Compute for any given x: 1 way/not reversible
- H(x): Collision free: 2 diff input values will result in diff hash values
- If hash hard to invert: 1-way hash: Computationally infeasible to find input for x such that h=H(x)

**Well-Known Hash Functions**

1. MD5 w/128-bit digest
2. SHA-256 w/256-bit digest

**MD5:** Legacy: Hashing alg: Ron Rivest: Used in variety of Internet apps today

- 1-way func that makes it easy to compute hash from given input data
- Diff to compute input data given only hash value
- Complex seq of simple bin ops [exclusive OR (XOR)/rotations] performed on input data
- Produce 128-bit hashed msg digest

**SHA: Secure Hash Alg:** NIST: Specified in **SHS: Secure Hash Standard**

- SHA-1: Legacy 1994: Corrected unpublished flaw in SHA
- Design similar to MD5
- Takes msg less than 2^64 bits in length: Produces 160-bit msg digest
- Alg slightly slower than MD5: Larger msg digest makes it more sec against brute-force collision/inversion attacks

**SHA-2: SHA-224/256/384/512:** Secure hash algs U.S. Gov requires by law for use in certain apps

- Includes use in other crypt algs/protocols for protection of sensitive unclassified info

**HMAC/KHMAC: Keyed-Hash Msg Auth Code:** Use addl secret key as input to hash: Adds auth to integrity assurance

- HMAC calc using specific alg combines hash w/secret key
- Only sender/receiver know key: Output of hash depends on input data/key
- Defeats MITM attacks/provides auth of data origin

**Cisco uses 2 HMAC funcs:**

| | |
|---|---|
| **Keyed MD5** | HMAC-MD5: Based on MD5: Marginal sec: Only when no alts avail |
| **Keyed SHA-1** | HMAC-SHA-1: Based on SHA-1: Provides more adequate sec |

**Key length**: AKA Key size: Measure in bits
**Keyspace:** # of possibilities that can be generated by specific key length: As key length increase:  Keyspace increases

- Keyspace of alg: Set of all possible key values

**Types of Crypto Keys:**

| | |
|---|---|
| **Symmetric** | Can be exchanged bet 2 rtrs supporting VPN |

| | |
|---|---|
| **Asymmetric** | HTTPS apps |
| **Digital sigs** | Connecting to sec website |
| **Hash** | Symmetric/asymmetric key gen/digital sigs/other apps |

**2 classes of encryption alg:**

| | |
|---|---|
| **Symmetric** | Algs use same pre-shared key to encrypt/decrypt data<br>• Pre-shared key: Known by sender/receiver before any encrypted comms |
| **Asymmetric** | Algs use diff keys to encrypt/decrypt data<br>• Sec msgs can be exchanged w/out having to have pre-shared key<br>• Neither party has shared secret: Very long key lengths used |

**Symmetric Encryption Summary**

| **Symmetric Encryption Alg** | **Key Length** |
|---|---|
| **DES** | 56 |
| **3DES** | 112/168 |
| **AES** | 128/192/256 |
| **SEAL: SW Encryption Alg** | 160 |
| **RC** | RC2: 40/64 \| RC4: 1-256 \| RC5: 0-2040 \| RC6: 128/192/256 |

**Symmetric Block/Stream Ciphers**

| | |
|---|---|
| **Block Ciphers** | Transform fixed-length block of plaintext into common block of ciphertext of 64/128 bits<br>• **Block size:** How much data encrypted at any 1 time<br>• DES w/64-bit block \| AES w/128-bit block |
| **Stream Ciphers** | Encrypt plaintext 1 byte/bit at time: Block cipher w/block size of 1 bit<br>• Transformation of smaller plaintext units varies<br>• Can be much faster than block: Generally don't increase msg size: Can encrypt arbitrary # of bit<br>• Example: Vigenère cipher<br>• A5 (used to encrypt GSM cell comm) \| RC4 \| DES |

**DES Symmetric Encryption**

**DES: Data Encryption Standard:** Legacy: Ops in block mode by encrypting data in 64-bit blocks

- Seq of permutations/substitutions of data bits combined w/encryption key
- Same alg/key used for encryption/decryption
- Has fixed key length: Key is 64-bits long: Only 56 bits used for encryption
- Remaining 8 bits used for parity to verify key's integrity: Last bit of each key byte used to indicate odd parity
- Always 56 bits long

**3DES Op: 3DES-Encrypt-Decrypt-Encrypt: 3DES-EDE: to encrypt plaintext**

- More effective at increasing sec than simply encrypting data 3x w/3 diff keys
- Provides encryption w/effective key length of 168 bits
- If K1/K3 eq: Less secure encryption of 112 bits achieved
- Resource intensive

**AES:** 1997: NIST: Rijndael block cipher as AES alg: Joan Daemen/Vincent Rijmen: Var block/key length

- Rijndael: Iterated block cipher: Initial input block/cipher key undergo multiple transformation cycles before output
- Alg can op over var-length block using var-length keys
- 128/192/256-bit key can be used to encrypt data blocks that are 128/192/256 bits long

Written so that block/key length/both can easily be extended in multiples of 32 bits

**SEAL: SW-Optimized Encryption Alg:** Alt alg to SW-based DES/3DES/AES

- Phillip Rogaway/Don Coppersmith: 1993: Stream cipher uses 160-bit key
- Lower impact on CPU compared to other SW-based algs: Longer initialization phase

**SEAL restrictions:**

- Cisco rtr/peer must support IPSec
- Cisco rtr/peer must run IOS img that supports encryption
- IOS images ID'd w/str "k9" in IOS filename
- Rtr/peer must not have HW IPsec encryption

**RC Algorithms: Ronald Rivest:**

|  |  |
|---|---|
| **RC2** | Var key-size block cipher designed as "drop-in" replacement for DES |
| **RC4** | Most widely used stream cipher: Var key-size Vernam stream cipher<br>• Often used in file encryption/sec comm: SSL<br>• Not considered OTP b/c key not random<br>• Cipher can run very quickly in SW |
| **RC5** | Fast block cipher that has var block size/key length: Drop-in replacement for DES if block size set to 64-bit |
| **RC6** | Dev: 1997: 128-256-bit block cipher designed by Rivest/Sidney/Yin: Based on RC5<br>• Designed to meet req of AES |

**Diffie-Hellman:** Whitfield Diffie/Martin Hellman: 1976

- Basis of most modern auto key exchange methods
- Not encryption mechanism/not typically used to encrypt data
- Used to securely exchange keys that encrypt data

Allows 2 computers to generate identical shared secret on both sys w/out comm before

- Commonly used: Data exchanged using IPsec VPN: Data encrypted on Internet using SSL/TLS/w/SSH data

**Asymmetric Key Algs:** AKA Public-key algs: Designed so key used for encryption diff from key used for decryption

- Decryption key can't be calc from encryption key/vice versa

**4 protocols that use asymmetric key algs:**

|  |  |
|---|---|
| **IKE** | Internet Key Exchange: Fundamental component of IPsec VPNs |
| **SSL** | Secure Socket Layer: Implemented as IETF standard TLS |
| **SSH** | Protocol that provides sec remote access connection to network devices |
| **PGP** | Pretty Good Privacy: Program that provides crypto privacy/auth: Often used to increase sec of email comm |

**Asymmetric algs use 2 keys:**

|  |  |
|---|---|
| Public | Private |

Both keys capable of encryption process: Complementary matched key required for decryption: Confidentiality/auth/integrity

**Characteristics:** Typical length: 512-4,096 bits

- Length greater than/eq to 1,024: Trusted
- Lengths shorter than 1,024 bits: Unreliable for most algs

**Public Key (Encrypt) + Private Key (Decrypt) = Confidentiality**

- When public key used to encrypt data: Private key must be used to decrypt data
- Only 1 host has private key; therefore, confidentiality achieved
- If private key compromised: Another key pair must be generated to replace compromised key

**Private Key (Encrypt) + Public Key (Decrypt) = Authentication**

- When encryption process started w/private key

**Asymmetric Algs**

| **Asymmetric Encryption Alg** | **Key Length (Bits)** | **Description** |
|---|---|---|
| **DH** | 512/1024/2048/3072/4096 | Public key: 1976:<br>• Allows 2 parties to agree on key to encrypt msgs sent<br>• Easy to raise # to certain power<br>• Diff to compute which power used given outcome |
| **DSS: Digital Sig Standard**<br>**DSA: Digital Sig Alg** | 512-1024 | DSS: NIST: Specifies DSA as alg for digital sigs<br>• Public key: Based on ElGamal sig scheme<br>• Sig creation speed similar w/RSA<br>• 10-40x as slow for verification |
| **RSA Encryption Algs** | 512-2048 | Rivest/Shamir/Adleman: MIT: 1977<br>• Public key: Based on diff of factoring large #'s<br>• 1st alg known suitable for signing/encryption<br>• Widely used in electronic commerce protocols |

| | | |
|---|---|---|
| **ElGamal** | 512-1024 | Public key: Based on DH key agreement<br>• Tahar ElGamal: 1984<br>• GNU Privacy Guard/PGP/cryptosystems<br>• Disadvantage: Encrypted msg becomes big |
| **Elliptical Curve Techniques** | 160 | Neil Koblitz/Victor Miller: Mid-80's<br>• Can adapt many crypto algs like DH/ElGamal<br>• Advantage: Keys much smaller |

**Digital Sigs:** Math technique used to provide 3 basic sec services: Specific properties: Enable entity auth/data integrity

**Common Use 2 situations:**

1. **Code signing:** Verify integrity of exe files DL from vendor: Auth/verify ID site
2. **Digital certs:** Verify ID of org to auth vendor site/establish encrypted connection to exchange confidential data

**Code Signing:** Commonly used to provide assurance of authenticity/integrity of SW codes

- Answers question: "How can usrs trust code DL from Internet?"
- Exe files wrapped in digitally signed envelope: Allows end usr to verify sig before installing SW

**Digitally signing code provides assurances about code:**

- Code auth/sourced by publisher
- Code hasn't been mod since it left SW publisher
- Publisher undeniably published code: Nonrepudiation of act of publishing

**Digital Certs:** Equiv of electronic passport

- Enable usrs/hosts/orgs to sec exchange info over Internet
- Used to auth/verify a usr sending a msg is who they claim to be
- Can be used to provide confidentiality for receiver w/means to encrypt reply

**Digital Sig Algs**

**3 DSS: Digital Sig Standard algs used for generating/verify digital signs:**

| | |
|---|---|
| **DSA** | Digital Sig Alg: Original standard for generating pub/priv key pairs: Generating/verifying digital sigs |
| **RSA** | Rivest-Shamir Adelman Alg: Asymmetric: Commonly used for generating/verifying digital sigs |
| **ECDSA** | Elliptic Curve Digital Sig Alg: Newer variant of DSA: Digital sig auth/non-repudiation<br>• Benefits of computational efficiency/small sig sizes/min BW |

**Digitally Signed Cisco SW**

- Digitally signed IOS imgs for many network devices like ISR rtrs
- Recognized by SPA char str contained w/in filename

**Each character of SPA has following meaning:**

| | |
|---|---|
| **S** | Digitally signed SW |
| **P** | Production img |
| **A** | Indicates key ver used to digitally sign img |

To verify digitally signed img on an ISR: sh software authenticity

**PKI: Public Key Infrastructure Overview:** Framework used to sec exchange info bet parties

- Foundation of PKI ID's cert authority analogous to licensing bureau
- CA issues digital certs that auth ID of org/usrs
- Certs used to sign msgs to ensure they haven't been tampered w/

**How does PKI work?**

| | |
|---|---|
| **PKI Framework** | Needed to support large-scale distribution/ID public keys<br>• Enables usrs/computers to sec exchange data over Internet/verify ID of other party<br>• PKI ID's encryption algs/lvl of sec/distribution policy to usrs<br>• Any form of sensitive data exchanged over Internet reliant on PKI for sec<br>○ Without: Confidentiality can still be provided but auth not guaranteed<br>• Framework consists of HW/SW/people/policies/procedures<br>○ Needed to create/manage/store/distribute/revoke digital certs |

**Cert Authorities:** Many vendors provide CA servers as managed service/end-user product:

- CA's: Can issue certs of # of classes: Determine how trusted it is

| **Class** | **Description** |
|---|---|
| **0** | Testing purposes when no checks performed |

| | |
|---|---|
| **1** | Individuals w/focus on verification of email |
| **2** | Orgs for which proof of ID req |
| **3** | Servers/SW signing for independent verification/checking of ID/authority done by issuing CA |
| **4** | Online business transactions bet companies |
| **5** | Private orgs/gov sec |

**Interoperability of Diff PKI Vendors:** IETF formed PKIX workgroup: PKI X.509: Promoting/standardizing PKI on Internet

**PKCS: Public-Key Crypto Standards:** Refers to group of standards devised/published by RSA Labs

- Basic interoperability of apps that use pub-key crypto: Defines low-lvl fmts for sec exchange of arbitrary data

**SCEP: Simple Cert Enrollment Protocol:** Need for cert mgmt protocol for PKI clients/CA servers

- Clients/servers can support cert lifecycle ops: Cert enrollment/revocation/CRL access

**After CA receives req: Can perform 1 of 3 functions:**

1. Auto approve req
2. Send cert back
3. Compel end entity to wait until op can manually auth ID of req end entity

**PKI Topologies:** Can form diff topologies of trust: Simplest single-root PKI topology

- Single CA: AKA root CA: Issues all certs to end usrs: Usually w/in same org
- Benefit: Simplicity: Diff to scale to large env

Larger networks: PKI CAs may be linked using 2 basic arch:

| | |
|---|---|
| **Cross-certified** | **Peer-to-peer model**<br>• Individual CAs establish trust relationships w/other CAs by cross-certifying CA certs<br>• Usrs in either CA domain also assured they can trust each other<br>• Provides redundancy/eliminates single-point of failure |
| **Hierarchical** | **Highest lvl CA:** Root CA: Can issue certs to end usrs/subordinate CA<br>• Sub-CAs could be created to support various domains/communities of trust<br>• Root CA maintains "community of trust" by ensuring each entity conforms to min practices<br>• Benefits: Increased scalability/manageability<br>• Works well in large orgs: Can be diff to determine chain of signing process |

**RA: Registration Authority:** Hierarchical: RA can accept reqs for enrollment in PKI

- Responsible for ID/Auth of subs but doesn't sign/issue certs

**3 specific tasks:**

1. Auth of usrs when they enroll w/PKI
2. Key gen for usrs that can't their own
3. Distribution of certs after enrollment

**Digital Certs/CAs**

1. In CA auth procedure: 1st step when contacting PKI is to obtain copy of public key of CA
    1. Public key verifies all certs issued by CA/vital for proper op of the PKI
2. Public key: AKA Self-signed cert: Also distributed in form of cert issued by CA: Only root CA issues self-signed certs
3. CA certs retrieved in-band over network: Auth done out-of-band using phone

Post 8

Thursday, January 24, 2019 11:33 PM

# VPNS: RTR/FW SEC CH 8

**VPN: Private Network** created over a public network (usually Internet)

- Instead of dedicated phys connection: Uses virtual connections routed through Internet from org to remote site
- Originally strictly IP tunnels that didn't include auth/encryption

**GRE: Generic Routing Encapsulation:** Tunneling protocol: Cisco:

- Can encapsulate variety of network layer protocol packet types inside IP tunnels
- Creates virtual point-to-point link to Cisco rtrs at remote points over IP internetwork

**Benefits:**

| | |
|---|---|
| **Cost Savings** | Reduces connectivity costs: Simultaneously increases remote connection BW |
| **Sec** | Advanced encryption/auth protocols protect data |
| **Scalability** | Allow orgs Internet use: Easy to add new usrs w/out adding infrastructure |
| **Compatibility** | Can be implemented across variety of WAN link options |

**L3 IPsec VPNs**

**VPN: Connects 2 endpoints:** 2 remote offices, over public network to form logical connection

- Logical connections can be made at L2 or L3
    - Examples of L3 VPNs: GRE/MPLS/IPSec
        - Can be point-to-point [GRE/IPSec]
        - Can establish any-to-any connectivity to many sites using MPLS

**IPSec: Suite of protocols dev w/backing of IETF to achieve sec services over IP packet-switched networks**

- Allow for auth/integrity/access control/confidentiality
- Info exchanged bet remote sites can be encrypted/verified
- Both remote-access/site-to-site VPNs can be deployed using IPsec

**2 Types of VPNs**

| | |
|---|---|
| **Remote-access** | Created when VPN info not statically set up: Allows for dynamically changing connection info<br>• Can be enabled/disabled when needed |
| **Site-to-site** | Created when devices on both sides of VPN aware of config in advance<br>• VPN remains static: Internal hosts have no knowledge it exists |

**Components of Site-to-Site VPNs**

- Hosts send/receive normal TCP/IP traffic through VPN GW (can be a rtr/fw/concentrator/etc..)
- VPN GW responsible for encapsulating/encrypting outbound traffic from particular site
    - Sending it through VPN tunnel over Internet to peer VPN GW at the other site
    - Upon receipt: Peer VPN GW strips headers/decrypts content/relays packet toward target host inside

| | |
|---|---|
| **MPLS VPN** | **<u>Set of sites interconnected by MPLS provider</u>**<br>• Each site 1/more CE: Customer Edge devices attach to 1 PE: Provider Edge devices<br>• Easier to manage/expand than conventional VPNs<br>• When new site added: Only SP's edge device that provides services needs to be updated |
| **DMVPN** | **<u>Enables auto-provisioning of site-to-site IPsec VPNs</u>**<br>• **<u>Combines 3 IOS features:</u>**<br>○ NHRP: Next Hop Resolution Protocol<br>○ Multipoint GRE: Generic Routing Encapsulation<br>○ IPsec VPN<br>• This combo eases provisioning challenges/provides sec connectivity bet locations |

| | |
|---|---|
| **GETVPN** | **Uses trusted group to eliminate point-to-point tunnels/associated overlay routing**<br>• All GMs: Group Members share common SA: Sec Association: AKA Group SA<br>• Enables GMs to decrypt traffic encrypted by any other GM<br>• No need to negotiate point-to-point IPsec tunnels bet members of a group<br>• GET VPN is "tunnel-less" |

## Hairpinning/Split Tunneling

**Hairpinning:** When VPN traffic that enters an int may also be routed out same int
**Split tunneling:** When VPN traffic must be split bet traffic destined for subnets (trusted)/to Internet (untrusted)

## IPsec Tech

- IETF standard: RFC 2401-2412: Defines how VPN can be sec across IP networks
- IPsec protects/auth packets bet source/destination
- Can protect virtually all traffic from L4-L7

**IPsec provides essential security functions:**

1. Confidentiality using encryption
2. Integrity using hashing algs
3. Auth using IKE: Internet Key Exchange
4. Sec key exchange using DH alg

| Protocol | Framework |
|---|---|
| **IPsec Protocol** | **AH: Authentication Header**<br>**ESP: Encapsulation Security Protocol**<br>• AH auths L3 packet<br>• ESP encrypts L3 packet |
| **Confidentiality** | **Encryption:**<br>• DES: Data Encryption Standard<br>• 3DES: Triple DES<br>• AES: Advanced Encryption Standard<br>• SEAL: SW-Optimized Encryption Alg |
| **Integrity** | **Ensures data arrives unchanged using hash alg:**<br>• MD5: Msg-digest 5<br>• SHA: Secure Hash Alg |
| **Authentication** | **IKE: Internet Key Exchange** to auth usrs/devices that can carry out comm independently<br>• IKE: Usrname/passwd/OTP/biometrics/PSKs: Pre-Shared Keys/Digital certs using RSA alg |
| **Diffie-Hellman** | Uses DH alg to provide pub key exchange method for 2 peers to establish shared secret key<br>• DH14/15/16<br>• DH 19/20/21/24<br>• DH1/2/5: No longer recommended |

**Confidentiality:** Achieved by encrypting data: Degree of sec depends on length of key used in encryption alg
**Integrity:** Data received is exactly same data that was sent

- B/C VPN data transported over public Internet: Method of proving data integrity is req
- HMAC: Data integrity alg: Guarantees integrity of msg using hash value

**Auth:** Device on other end of tunnel must be auth before comm path is considered sec

## Secure Key Exchange

- Encryption algs req symmetric, shared secret key to perform encryption/decryption
- Easiest key exchange method is to use pub key exchange method: DH

**Variations of DH key exchange specified as DH groups:**

| | |
|---|---|
| **DH groups 1/2/5** | Support exponentiation over prime modulus w/key size of 768/1024/1536 bits: Not recommended |
| **DH groups 14/15/16** | Key sizes w/2048/3072/4096 bits: Recommended until 2030 |
| **DH groups 19/20/21/24** | Key sizes 256/384/521/2048 bits support ECC: Elliptical Curve Crypto<br>• Reduces time needed to gen keys |

| | |
|---|---|
| | • 24 is preferred |

DH group chosen must be strong/have enough bits, to protect IPsec keys during negotiation
**RFC 4869:** Defines set of crypto algs to adhere to NSA standards for classified info:
**Suite B: Includes these specified algs:**

- Encryption should use AES 128/256-bit keys
- Hashing should use SHA-2
- Digital Sigs should use ECDSA: Elliptic Curve Digital Sig Alg w/256/384-bit prime moduli
- Key exchange should use Elliptic Curve DH

## IPsec Protocol Overview

**2 main IPsec protocols:**

1. **AH: Auth Header**
2. **ESP: Encapsulation Security Protocol**

IPsec protocol 1st building block of framework: Choice of AH/ESP establishes which other building blocks avail

| | |
|---|---|
| **AH** | IP protocol 51<br>• Appropriate only when confidentiality not required/permitted<br>• Provides data auth/integrity<br>• Doesn't provide confidentiality (encryption)<br>• All txt transported unencrypted |
| **ESP** | IP protocol 50<br>• Provides both confidentiality/auth<br>• Performs encryption on IP packet<br>• Provides auth for inner IP packet/ESP header<br>• Auth provides data origin auth/data integrity |

## Authentication Header

- Achieves authenticity by applying a keyed 1-way hash to packet to create hash/msg digest
- Hash combined w/txt/transmitted in plaintext
- Receiver detects changes in any part of packet that occur during transit
  - Performs same 1-way hash on received packet/compares result to value of msg digest sender supplied
- Authenticity assured b/c 1-way hash also employs shared secret key bet the 2 sys

**AH function applied to entire packet:** Except any IP header fields that normally change in transit

- Mutable fields: Fields that normally change during transit: Example: TTL

**Process:**

1. IP header/data payload are hashed using shared secret key
2. Hash builds new AH header: It's inserted into original packet
3. New packet transmitted to IPsec peer router
4. Peer rtr hashes IP header/payload using shared secret key: Extracts transmitted hash from AH header/compares
   - Hashes must match exactly: If 1 bit changed in transmitted packet:
   - Hash output on received packet changes/AH header won't match

AH may not work if the environment uses NAT.

## ESP: Encapsulated Security Protocol

- Provides confidentiality by encrypting payload
- Supports variety of symmetric encryption algs
- If ESP selected: Encryption alg must also be selected: Default: 56-bit DES
- Can also provide integrity/auth
- Payload is encrypted: Encrypted payload sent through hash alg: Hash provides auth/data integrity for payload

**ESP can also enforce anti-replay protection:**

- Anti-replay protection verifies each packet is unique/not duplicated
- Keeps track of packet seq #'s/using sliding win on dest end
- Anti-replay used in ESP, but supported in AH

**ESP Encrypts/Auth**

- Ability to protect original data b/c entire original IP datagram/ESP trailer encrypted

- ESP auth: Encrypted IP datagram/trailer/ESP header included in hashing process
- New IP header attached to auth payload: New IP used to route packet through Internet

**When both auth/encryption selected: Encryption performed 1st**

- Order of processing
- Facilitates rapid detection/rejection of replayed/bogus packets by receiving device
- Prior to decrypting packet: Receiver can auth inbound packets: Quickly detect problems/DoS attacks

## Transport/Tunnel Modes

**ESP/AH can be applied to IP packets in 2 diff modes:**

1. Transport
2. Tunnel

| | |
|---|---|
| **Transport Mode** | Sec provided only for transport layer of OSI/above<br>• Protects payload of packet but leaves original IP in plaintext<br>• Original IP used to route packet through Internet<br>• ESP transport mode used bet hosts |
| **Tunnel Mode** | Provides sec for complete original packet<br>• Original packet encrypted/encapsulated in another packet: AKA: IP-in-IP encryption<br>• IP on outside packet used to route packet through Internet |

**IKE: Internet Key Exchange Protocol**

- Key mgmt protocol standard
- Used in conjunction w/IPsec
- Auto negotiates IPsec security associations/enables IPsec secure comm
- Enhances IPsec by adding features/simplifies config for standard
- Without IKE: IPsec config would be complex/manual config process that would not scale well

**Hybrid protocol:** Implements key exchange inside ISAKMP: Internet Sec Association Key Mgmt Protocol framework

- ISAKMP defines msg fmt/mechanics of key exchange protocol/negotiation process to build an SA for IPsec
- IKE implements portions of the Oakley/SKEME protocols but not dependent on them

**Instead of transmitting keys directly across network:**

- IKE calcs shared keys based on exchange of series of data packets
- Disables a 3rd party from decrypting keys even if they capture all exchanged data that was used to calc the keys
- UDP port 500 to exchange IKE info bet sec GW
- Port 500 packets must be permit on any IP int connecting a sec GW peer

## Phase 1/2 Key Negotiation: IKE uses ISAKMP for phase 1/2 of key negotiation

| | |
|---|---|
| **Phase 1** | Negotiates sec association (key) bet 2 IKE peers<br>• Key negotiated in phase 1 enables IKE peers to comm securely in P2<br>• During P2 negotiation: IKE establishes keys (sec associations) for other apps such as IPsec<br>**In P1:** 2 IPsec peers perform initial negotiation of SAs<br>• **Purpose:** Negotiate ISAKMP policy/auth peers/set up sec tunnel bet them<br>• Tunnel will be used in Phase 2 to negotiate IPsec policy<br>**Can be implemented in main/aggressive mode:**<br>**Main mode:** ID's of 2 IKE peers hidden<br>**Aggressive mode:** Less time than main to negotiate keys bet peers<br>○ Since auth/hash sent unencrypted before tunnel established: Aggressive mode vuln to brute-forces |
| **Phase 2** | Negotiating Sas<br>• Purpose is to negotiate IPsec sec params to be used to secure IPsec tunnel<br>• Quick mode: Can only occur after IKE established sec tunnel in Phase 1<br>• SAs negotiated by IKE process ISAKMP on behalf of IPsec: Needs encryption keys for op<br>• SAs that IPsec uses are unidirectional; therefore, separate key exchange req for each data flow<br>• Quick mode also renegotiates new IPsec SA when IPsec SA lifetime expires |

**IPsec Negotiation**

1. ISAKMP tunnel is initiated when host A sends "interesting" traffic to host B.

    - Traffic considered interesting when it travels bet peers/meets criteria defined in an ACL
2. IKE P1: Peers negotiate ISAKMP SA policy: When peers agree on policy/auth: Sec tunnel created
3. IKE P2: IPsec peers use tunnel to negotiate SA policy: Determines how tunnel is established
4. IPsec tunnel created: Data transferred bet IPsec peers based on SA's
5. IPsec tunnel terminates when SAs manually deleted/their lifetime expires

**Permit isakmp traffic:**

**R1(config)# access-list acl permit udp source wildcard destination wildcard eq isakmp**

**Permit esp traffic:**

**R1(config)# access-list acl permit esp source wildcard destination wildcard**

**Permit ah traffic:**

**R1(config)# access-list acl permit ahp source wildcard destination wildcard**

**show crypto isakmp default policy** View default policies

**crypto isakmp policy** Config new iksamp policy

- Only arg for cmd is to set priority for policy from 1-10000
- Peers will attempt to negotiate using policy w/lowest # (highest priority)
- Peers don't req matching priority #'s

**When in ISAKMP policy config mode: SAs for IKE P1 can be config: 5 SA's to config:**

| | | | | |
|---|---|---|---|---|
| Hash | Auth | Group | Lifetime | Encryption |

**show crypto isakmp policy Verify config**

**Config Pre-Shared Key**

**R1(config)# crypto isakmp key keystring address peer-address**

**R1(config)# crypto isakmp key keystring hostname peer-hostname**

**show crypto isakmp sa**

**crypto ipsec transform-set Config IPsec Transform Set**

**Config Crypto Map**

**R1(config)# crypto map map-name seq-num [ipsec-isakmp | ipsec-manual]**

| | |
|---|---|
| **map-name** | ID crypto map set |
| **seq-num** | Seq # assigned to crypto map: **crypto map map-name seq-num** w/out any keyword to mod existing crypto map |
| **ipsec-isakmp** | IKE will be used to establish IPsec for protecting traffic specified by crypto map |
| **ipsec-manual** | IKE won't be used to establish IPsec SA's for protecting traffic specified by crypto map |

**Crypto Map Config**

1. Bind ACL: Transform set to map
2. Specify peer's IP
3. Config DH group
4. Config IPsec tunnel lifetime

**show crypto map** Verify crypto map config

**Apply Crypto Map**: Enter int config: Outbound int: Config **crypto map map-name**

**Send Interesting Traffic:** Test 2 tunnels by sending interesting traffic across link: When traffic matches ACL config on both rtrs

**Verify ISAKMP/IPsec Tunnels**

**show crypto isakmp sa**

**show crypto ipsec sa**

# Post 9

Thursday, January 24, 2019 11:33 PM

# ASA P2

**ASA 5505:** Basic traffic filtering w/ACLs: Many similarities bet ASA ACLs/IOS ACLs

**ASA ACLs differ from IOS ACLs:** Use a network mask instead of wildcard mask: ASA: Named instead of #'d

| Sim/Diff | Rules/Features |
|---|---|
| **ASA/IOS** | **ACLs: Made up of 1/more ACEs:** ACEs applied to protocol/source-dest IP/network/source-dest ports<br>• Processed sequentially from top down<br>• Criteria match will cause ACL to exit<br>• Implicit deny any at bottom<br>• Remarks can be added per ACE or ACL<br>• Only apply 1 access list per int/protocol/direction<br>• Can be enabled/disabled based on time ranges |
| **ASA Only** | Uses network mask: Not wildcard mask [0.0.0.255]<br>• Always named instead of #'d<br>• Default: Int sec levels apply access control w/out ACL config |

**Types of ASA ACL Filtering:** ACLs on sec app can be used to filter packets passing through it/also packets destined for app

| | |
|---|---|
| **Through-traffic** | Traffic passing through the sec appliance from 1 int to another<br>Config complete in 2 steps: 1. Set up ACL 2. Apply ACL to int |
| **To-the-box-traffic** | **AKA: Mgmt Access Rule:** Applies to traffic that terms on ASA<br>• Introduced in ver 8 to filter traffic destined for control plane of ASA<br>• Completed in 1 step: Req addl set of rules to implement access control |

**ASA differs from rtr counterparts b/c of int sec lvls:** Default: Sec lvls apply access control w/out ACL config
**Allow connectivity bet ints w/same sec lvls:**
**same-security-traffic permit inter-interface** [global] **Required**
**Enable traffic to enter/exit same int (when encrypted traffic enters int/routed out same int):**
**same-security-traffic permit intra-interface** [global]
**Types of ASA ACLs**

| ACL Use | Description |
|---|---|
| **Control network access for IP traffic** | ASA doesn't allow any traffic from lower sec int to higher sec int<br>• Unless explicitly perm by an extended access-list |
| **ID Traffic for AAA rules** | AAA rules use ACL's to ID traffic |
| **ID addr for NAT** | Policy NAT lets you ID local traffic for add transl<br>• By specifying the source/dest addr in an extended acl |
| **Establish VPN access** | Extended ACL can be used in VPN cmds |
| **ID traffic for MPF: Mod Policy Framework** | ACL's can be used to ID traffic in a class map<br>• Used for features that support MPF<br>Features that support MPF: TCP/General connection settings/inspection |

**Standard ACL**

| ACL use | Description |
|---|---|
| **ID OSPF dest network** | Standard ACL's include only dest addr<br>• Can be used to control redistribution of OSPF routes |
| **IPv6: Control network access** | Can be used to add/apply ACL's to control traffic |

**ASA supports 5 types of ACL's:**

| | |
|---|---|
| **Extended** | Most common: Contains 1/more ACE's to specify source/dest addr/protocol/ports [tcp/udp]/ICMP type |
| **Standard** | Unlike IOS where standard ID's source host/network: **ASA: Standard used to ID dest IP**<br>• Used for OSPF/in route map for OSPF redistribution<br>• Standard ACL's can't be applied to ints to control traffic |
| **EtherType** | Config only if sec app running in transparent mode |
| **Webtype** | Used in config that supports filtering for clientless SSL VPN |
| **IPv6 ACL** | Used to determine which IPv6 traffic to block/fwd at rtr ints |

**help access-list** *[priv]* Display syntax for all ACL's supported on ASA

**ASA ACL Elements**

| Element | Description |
|---|---|
| **ACL ID** | Name of ACL: Alphanum up to 241 chars |
| **Action** | Permit/Deny |
| **Protocol # – Source** | Can be IP for all traffic/name/IP protocol # (0-250)<br>**icmp** [1]/**tcp**[6]/**udp**[17]/protocol object-group |
| **Source** | ID's source/can be **any**/**host**/network/network object group<br>• To-the-box traffic filtering? **int** keyword used to specify source int of ASA |
| **Source port op** | In conjunction w/source port [optional]<br>• Includes: lt/gt/eq/neq/range |
| **Source port** | Can be actual TCP/UDP port #/select port name/service object group |
| **Destination** | ID's dest: Can be any/host/network/network object group<br>• To-the-box: int keyword used to specify dest of int of ASA |
| **Log** | Can set elements for syslog including severity lvl/log interval |
| **Time range** | Specify time range for ACE |

**Applying ACLs:** After ACL configured: Apply to int in inbound/outbound direction

**Verify ACLs:**

**show access-list**

**show running-config access-list**

**Erase config ACL:**

**clear config access-list id**

**access-group syntax:**

**access-group id { in | out } int ifname [per-user-override | control-plane ]**

| Cmd | Description |
|---|---|
| **access-group** | Apply ACL to int |
| **id** | Name of ACL to be applied to int |
| **in** | Filter inbound packets |
| **out** | Filter outbound packets |
| **int** | Keyword to specify int to apply ACL |
| **ifname** | Name of int to apply ACL |
| **per-user-override** | Allows downloadable ACL's to override entries on ACL int |
| **control-plane** | Specify whether applied ACL analyzes traffic destined to ASA mgmt purposes |

**Examples:**

**ASA(config)# access-list ACL-IN extended permit ip any any**

**ASA(config)# access-group ACL-IN in interface inside**

- Allows all hosts on inside network to go through ASA
- Default: All other traffic denied unless explicitly perm

**ASA(config)# access-list ACL-IN extended deny tcp 192.168.1.0 255.255.255.0 209.165.201.0**

**255.255.255.224**
**ASA(config)# access-list ACL-IN extended permit ip any any**
**ASA(config)# access-group ACL-IN in interface inside**

- ACL prevents hosts on 192.168.1.0/24 from accessing 209.165.201.0/27 network
- Internal hosts perm access to all other addr
- All other traffic implicitly denied

**ASA(config)# access-list ACL-IN extended permit ip 192.168.1.0 255.255.255.0 209.165.201.0 255.255.255.224**
**ASA(config)# access-group ACL-IN in interface inside**

- ACL allows hosts on 192.168.1.0/24 to access 209.165.201.0/27 network
- Default: All other traffic denied unless explicitly perm

**ASA(config)# access-list ACL-IN extended deny tcp any host 209.165.201.29 eq www**
**ASA(config)# access-list ACL-IN extended permit ip any any**
**ASA(config)# access-group ACL-IN interface inside**

- Prevents all inside hosts access to web service at 209.165.201.29
- Internal hosts perm access to all other services at 209.165.201.29
- Internal hosts perm access to all other addr
- All other traffic implicitly denied

**ACLs & Object Groups**
**Verify ACL syntax**
**show running-config access-list**
**show access-list**

**Extended ACL Config**
**ASA(config)# access-list ACL-IN remark permit PC-1 -> Server A for HTTP / SMTP**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.1 host 209.165.202.131 eq http**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.1 host 209.165.202.131 eq smtp**

**ASA(config)# access-list ACL-IN remark Permit PC-1 -> Server B for HTTP / SMTP**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.1 host 209.165.202.132 eq http**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.1 host 209.165.202.132 eq smtp**

**ASA(config)# access-list ACL-IN remark Permit PC-2 -> Server A for HTTP / SMTP**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.2 host 209.165.202.131 eq http**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.2 host 209.165.202.131 eq smtp**

**ASA(config)# access-list ACL0IN remark Permit PC-2 -> Server B for HTTP / SMTP**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.2 host 209.165.202.132 eq http**
**ASA(config)# access-list ACL-IN extended permit tcp host 209.165.201.2 host 209.165.202.132 eq smtp**

**ASA(config)# access-list ACL-IN extended deny ip any any log**
**ASA(config)# access-group ACL-IN in interface outside**

**Verify ACL:**
**ASA(config)# show run access-list**
**ASA(config)# show access-list ACL-IN brief**

**ACL Using Object Groups:**
**Object grouping:** Group similar items together to reduce # of ACEs

- Grouping like objects together: Object groups can be used in ACL instead of enter ACE for each object

**Sec appliance follows multiplication factor rule when ACEs defined**

**Number of ACEs = (2 outside hosts) x (2 internal servers) x (2 services) = 8**

- Object grouping can cluster network objects into 1 group/outside hosts into another
- Sec app can also combine both TCP services into service object group

**After object groups have been config: Can be used in any ACL/multiple ACLs**

- Single ACE could be used to allow trusted hosts to make specific service req to group of internal servers
- Objects can be reused in other ASA cmds: Easily altered
- Object groups can be nested in other object groups

**Condensed Extended ACL Syntax w/Object Groups**

**access-list id extended** { **deny | permit** } **protocol object-group** *network-obj-grp-id* **object-group** *network-obj-grp-id* **object-group service-obj-grp-id**

```
ASA(config)# object-group network NET-HOSTS
ASA(config-network-object-group)# description OG matches PC-A and PC-B
ASA(config-network-object-group)# network-object host 209.165.201.1
ASA(config-network-object-group)# network-object host 209.165.201.2

ASA(config)# object-group network SERVERS
ASA(config-network-object-group)# description OG matches Web / Email Servers
ASA(config-network-object-group)# network-object host 209.165.202.131
ASA(config-network-object-group)# network-object host 209.165.202.132

ASA(config)# object-group service HTTP-SMTP tcp
ASA(config-network-object-group)# description OG matches SMTP / WEB traffic
ASA(config-network-object-group)# port-object eq smtp
ASA(config-network-object-group)# port-object eq www

ASA(config)# access-list ACL-IN remark Only permit PC-A / PC-B -> Internal Servers
ASA(config)# access-list ACL-IN extended permit tcp object-group NET-HOSTS object-group
SERVERS object-group HTTP-SMTP
```

**Verify ACL/Object Group Config:**

```
ASA(config)# show run access-list
```

**ASA NAT: ASA supports NAT:** Typically used to translate private IP network addr into public IP's

**ASA supports following common types of NAT:**

| | |
|---|---|
| **Dynamic NAT** | Many-to-many transl: Usually an inside pool of priv addr req public addr from another pool |
| **Dynamic PAT** | Many-to-1 transl: AKA NAT overloads: Usually inside pool of priv addr overloading outside int/addr |
| **Static NAT** | 1-to-1 transl: Usually outside addr mapping to internal server |
| **Policy NAT** | Based on set of rules: Can specify only certain source addr for specific dest addr<br>• Or specific ports to be transl |

**ASA NAT feature: Twice-NAT**

- Twice-NAT ID's both source/dest addr in single rule nat cmd
- Used when config remote-access IPsec/SSL VPNs

**Types of NAT Deployments**

| | |
|---|---|
| **Inside NAT** | When host from higher-sec int has traffic destined for lower-sec int<br>• ASA translates internal host addr into global one<br>• ASA then restores original inside IP for return traffic |
| **Outside NAT** | When traffic from lower-sec int destined for host on higher-sec int must be transl<br>• May be useful to make enterprise host located on outside of internal appear<br>• As 1 from known internal IP |
| **Bidirectional NAT** | Indicates both inside/outside NAT used together |

**Config Dynamic NAT:** Network object dynamic NAT: 2 network objects req:

1. Network object ID'ing pool of public IP's which internal addr translate
   - ID'd using **range/subnet network object** cmds
2. Network object ID's internal addr to be translated/binds 2 objects together

    - ID's using **range/subnet network object** cmds
    - 2 network objects bound using **nat (real-ifc, mapped-ifc) dynamic mapped-obj**network object cmd

**Verify NAT using:**
**show xlate**
**show nat**
**show nat detail**

**Dynamic NAT Config:**
**ASA(config)# object network PUBLIC**
**ASA(config-network-object)# range 209.165.200.240 209.165.200.248**
**ASA(config-network-object)# exit**
**ASA(config)# object network DYNAMIC-NAT**
**ASA(config-network-object)# subnet 192.168.1.0 255.255.255.224**
**ASA(config-network-object)# nat (inside, outside) dynamic PUBLIC**

**Enable Return Traffic:**
**ASA(config)# policy-map global_policy**
**ASA(config-pmap)# class inspection_default**
**ASA(config-cmap)# access-list ICMPACL extended permit icmp any any**
**ASA(config)# access-group ICMPACL in interface outside**

**Config Dynamic PAT**
- Variation of config called Dynamic PAT
- When actual external IP config/overloaded instead of ASA int IP
- Only 1 network object req when overloading outside int
- To enable inside hosts to overload outside addr:
    - **nat (real-ifc,mapped-ifc) dynamic** interface cmd

**Dynamic PAT Config Example:**
**ASA(config)# object network INSIDE-NET**
**ASA(config-network-object)# subnet 192.168.1.0 255.255.255.224**
**ASA(config-network-object)# nat (inside,outside) dynamic interface**

**Config Static NAT**
- Config when inside addr mapped to outside addr
- **nat (real-ifc,mapped-ifc) static mapped-inline-host-ip** network object cmd

**clear nat counters** For testing NAT

**Config DMZ Int**
**ASA(config)# int Vlan3**
**ASA(config-if)# no forward int Vlan1**
**ASA(config-if)# nameif dmz**
**ASA(config-if)# security-level 70**
**ASA(config-if)# ip address 192.168.2.1 255.255.255.0**

**ASA(config)# int e0/2**
**ASA(config-if)# switchport access vlan3**
**ASA(config-if)# no shut**

**Static NAT Config Example:**
**ASA(config)# object network DMZ-SERVER**
**ASA(config-network-object)# host 192.168.2.3**
**ASA(config-network-object)# nat (dmz,outside) static 209.165.200.227**

**ASA(config)# access-list OUTSIDE-DMZ extended permit ip any host 192.168.2.3**
**ASA(config)# access-group OUTSIDE-DMZ in interface outside**

**ASA(config)# policy-map global_policy**
**ASA(config-pmap)# class inspection_default**
**ASA(config-pmap-c)# access-list ICMPACL extended permit icmp any any**

**ASA(config)# access-group ICMPACL in interface dmz**
**Local DB/Servers:** ASA can be config to auth using local usr db/external server for auth/both

- Local AAA uses local db for auth: Stores usrnames/passwds locally on ASA/usrs auth against local db
- Ideal for small networks that don't need dedicated AAA server
- ASA devices don't support local auth w/out using AAA

**Create local usr accts:**
**username name password password [*privilege priv-level*]**
**Erase usr from local db:**
**clear config username [name]**
**View all usr accts:**
**show running-config username**
**Server-based AAA auth: More scalable method than local:**

- Uses external db server resource leveraging RADIUS/TACACS+ protocols
- If multiple networking devices: Server-based AAA better

**Erase all AAA server configs:**
**clear config aaa-server**
**View all usr accs:**
**show running-config aaa-server**

**RADIUS/TACACS+ Server Cmds**

| **ASA Cmd** | **Description** |
|---|---|
| **aaa-server server-tag protocol protocol** | Creates TACACS+/RADIUS AAA server group |
| **aaa-server server-tag [int-name] host [server-ip\|name] key** | Configs AAA server as part of AAA server group<br>• Also configs AAA server params host-specific |

**ASA(config)# username Admin password class privilege 15**
**ASA(config)# show run username**

**ASA(config)# aaa-server TACACS-SVR protocol tacacs+**
**ASA(config-aaa-server-group)# aaa-server TACACS-SVR (dmz) host 192.168.2.3**
**ASA(config-aaa-server-host)# exit**
**ASA(config)# show run aaa-server**

**AAA Config:** Auth usrs who access ASA CLI over con SSH/HTTPS (ASDM)/Telnet connection
**aaa authentication console** [global]
**aaa authentication { serial | enable | telnet | ssh | http } console { LOCAL | server-group [ LOCAL ]}**
**Erase all AAA params:**
**clear config aaa**
**AAA Config:**
**ASA(config)# aaa authentication http console TCACS-SVR LOCAL**
**ASA(config)# aaa authentication enable console TACACS-SVR LOCAL**
**ASA(config)# aaa authentication http console TACACS-SVR LOCAL**
**ASA(config)# aaa authentication serial console TACACS-SVR LOCAL**
**ASA(config)# aaa authentication ssh console TACACS-SVR LOCAL**
**ASA(config)# aaa authentication telnet console TACACS-SVR LOCAL**

**ASA(config)# show run aaa**

**MPF: Modular Policy Framework:**

- Defines a set of rules for applying FW features: Such as traffic inspection/QoS/to traffic that traverses ASA
- Allows granular classification of traffic flows to apply diff advanced policies to diff flows
- Used w/HW modules to redirect traffic from ASA to modules that use Cisco MPF
- Can be used for advanced App Layer inspection of traffic by classifying at L5-L7
- Rate limiting/QoS can also be implemented

**Uses 3 config objects to define modular/object-oriented/hierarchical policies:**

- MPF syntax similar to ISR MQC: Modular QoS CLI | C3PL:  Common Classification Policy Lang
- Config params differ
- ASA provides more config actions compared to ISR for Cisco IOS ZPF
- ASA supports L5-L7 inspections using richer set of criteria for app-specific params

- ASA MPF feature can be used to match HTTP URLs/req methods:
    - Prevent usrs from surfing to specific sites during specific times
    - Prevent usrs from DL'ing music/vid files via HTTP/FTP or HTTPS/SFTP

**4 Steps to config MPF on ASA:**

1. Optional: Config extended ACLs to ID granular traffic that can be referenced in class map
2. Config class map to ID traffic
3. Config policy map to apply actions to those class maps
4. Config service policy to attach policy map to int

**MPF**

| | |
|---|---|
| **Classify Traffic: Class Maps** | ○ ID traffic on which to perform MPF<br>○ Create L3/4 class maps that can contain multiple match criteria<br>○ class-map class-name |
| **Define Actions: Policy Maps** | ○ Define a policy for traffic at L3-L7<br>○ Create policy map that can contain multiple class maps w/associated actions<br>○ policy-map policy-name |
| **Activate Policy: Service Policy** | ○ Activate policy map on ints<br>○ Create service policy that applies policy map to an int/all ints<br>○ service-policy serv-name<br>○ int intfname |

**Config Class Maps**

- Class maps config to ID L3/4 traffic
- To create class map/enter class-map config mode:
    - class-map class-map-name [global]
    - Names class-default/any name that begins w/_internal/_default reserved
    - Class map name must be unique: Can be up to 40 chars
- Traffic to match should be ID using match any/match access-list access-list-name

**Display info about class map config**
**show running-config class-map**
**Remove all class maps**
**clear configure class-map** [global]

**Class Maps ID Traffic:**

```
ASA(config)# access-list UDP permit udp any any
ASA(config)# access-list TCP permit tcp any any
ASA(config)# access-list SERVER permit ip any host 10.1.1.1

ASA(config)# class-map ALL-TCP
ASA(config-cmap)# description “This class-map matches all TCP traffic”
ASA(config-cmap)# match access-list TCP

ASA(config)# class-map ALL-UDP
ASA(config-cmap)# description “This class-map matches all UDP traffic”
ASA(config-cmap)# match access-list UDP

ASA(config)# class-map ALL-HTTP
ASA(config-cmap)# description “This class-map matches all HTTP traffic”
ASA(config-cmap)# match port TCP eq http

ASA(config)# class-map TO-SERVER
ASA(config-cmap)# description “Class map matches traffic   10.1.1.1”
ASA(config-cmap)# match access-list SERVER
```

**Define/Activate a Policy**

- Policy maps used to bind class maps w/actions
- policy-map policy-map-name [global] Apply actions to L3/4 traffic

**policy-map config mode** [config-pmap]:

| | |
|---|---|
| **class class-map-name** | ID specific class map on which to perform actions |

| | |
|---|---|
| | • Max # of policy maps: 64<br>• Can be multiple L3/4 class maps in 1 policy map<br>• Multiple actions can be assigned from 1/more feature types to each class map |

**3 most common cmds avail policy map config mode:**

| | |
|---|---|
| **set connection** | Sets connection values |
| **inspect** | Provides protocol inspection servers |
| **police** | Sets rate limits for traffic in this class |

**Display info about policy map config:**
**show running-config policy-map**
**Remove all policy maps:**
**clear configure policy-map** [global]

**Config Service Policy**
**Activate policy map globally on all ints/targeted int**
**service-policy** [global]
**service-policy policy-map-name [ global | interface intf ]**

**Implementing MPF:**

```
ASA(config)# access-list  TFTP-TRAFFIC permit udp any any eq 69
ASA(config)# class-map CLASS-TFTP
ASA(config-cmap)# match access-list TFTP-TRAFFIC

ASA(config)# policy-map POLICY-TFTP
ASA(config-pmap)# class CLASS-TFTP
ASA(config-pmap-c)# inspect tftp
ASA(config-pmap-c)# exit
ASA(config)# service-policy POLICY-TFTP global
```

**SA Default Policy**

- ASA default config includes a global policy that matches all default app inspection traffic
- Applies inspection to traffic globally
- Otherwise: Service policy can be applied to an int/globally

**Int service policies take precedence over global service policy for given feature**

- To alter global policy: Admin needs to edit default policy/disable default policy/apply new one

**Display info about service policy config**
**show service-policy**
**show running-config service-policy**
**Remove all service policies:**
**clear configure service-policy**
**Clear service policy statistics:**
**clear service-policy**