ICT2203/ICT2203X

Network Security

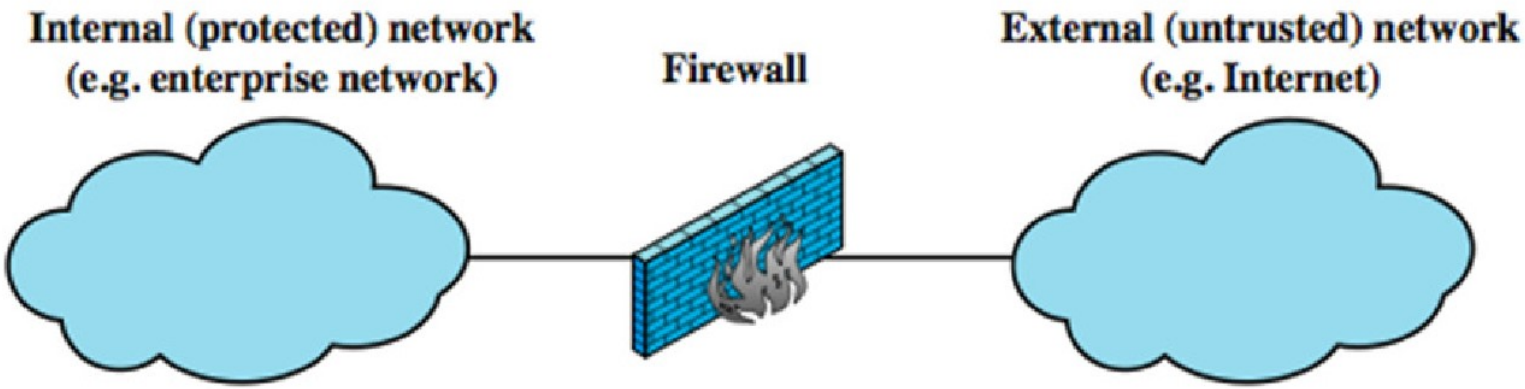
SINGAPORE INSTITUTE OF TECHNOLOGY

Lab 6 Notes:

Attacks and Defense of IP Networks with Firewalls

2021-2022 Trimester 3

Theoretically, a network firewall is a device or software that segments networks into different security zones, and enforces access control policy on traffic crossing between them.

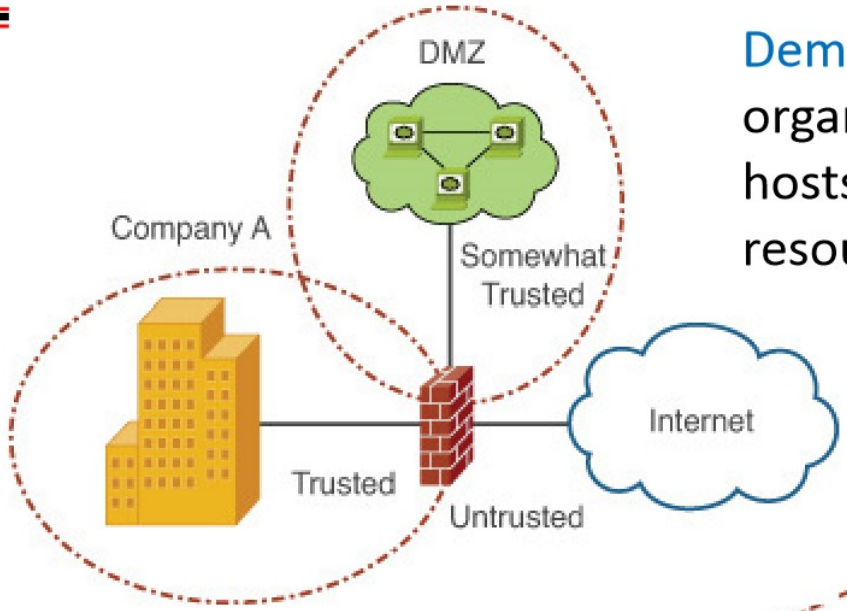


To be effective in enforcing access control policy, a firewall must be:

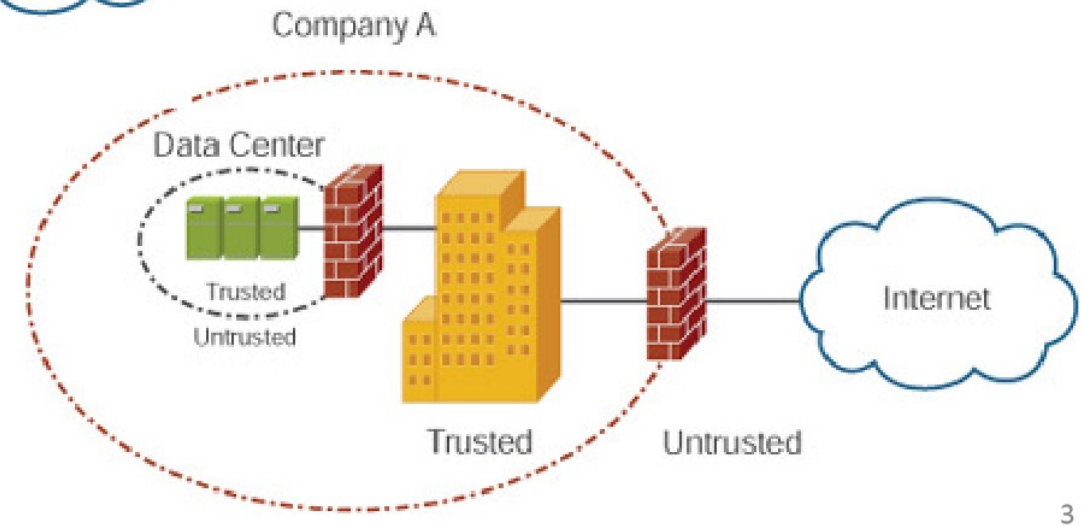
- Only transit point between different security zones
- Resistant to attacks itself

Limitation: Firewall cannot enforce access control policy on network traffic within the same security zone that do not pass through the firewall.

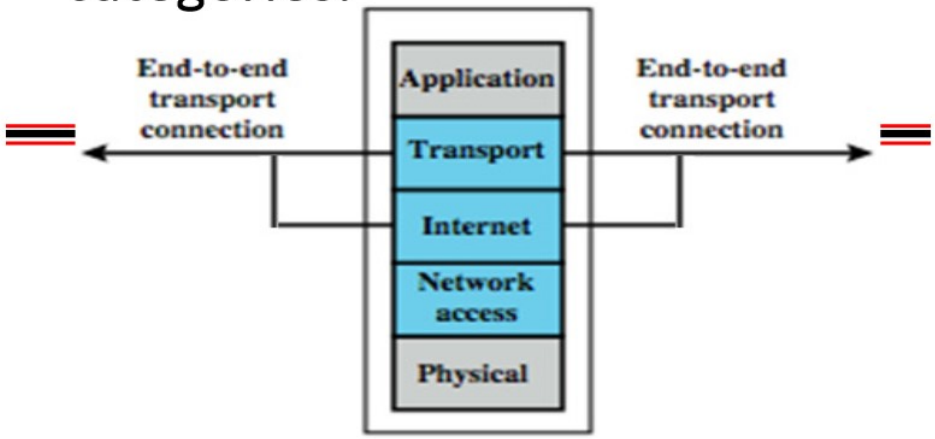
To segment networks into more than two security zones, an enterprise may implement one or more firewalls, e.g.:



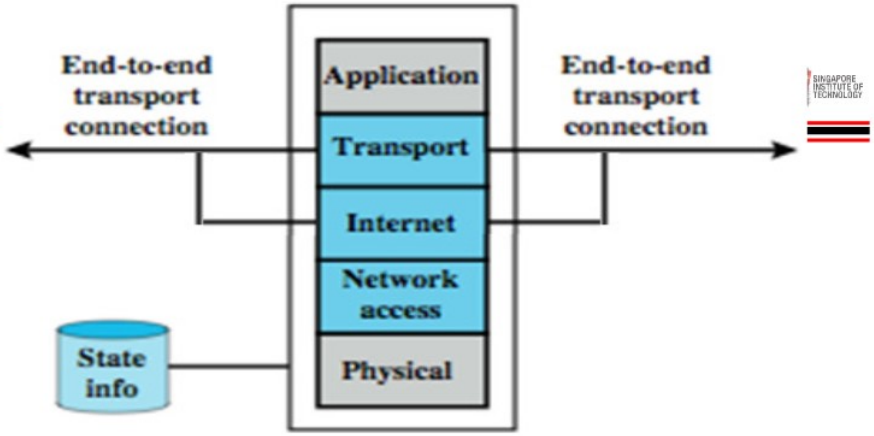
Demilitarized zone (DMZ) allows an organization to allow (untrusted) external hosts limited access to organization's resources, e.g. website.



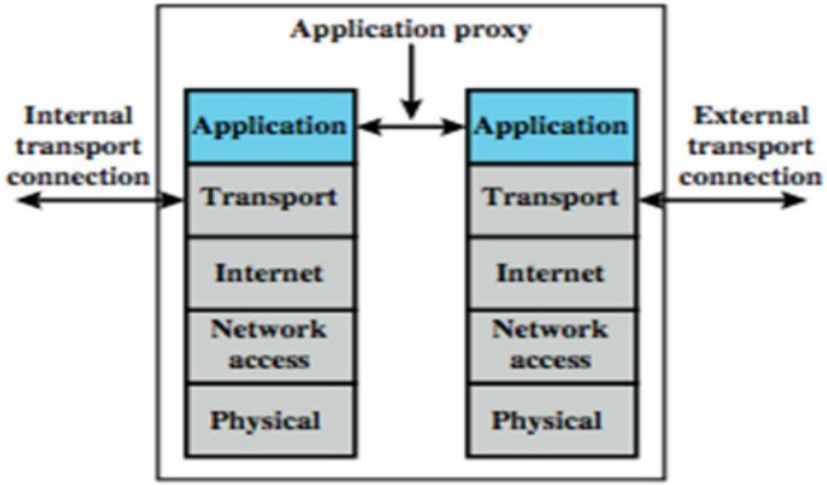
Broadly, firewall technologies may be divided into 4 different categories.



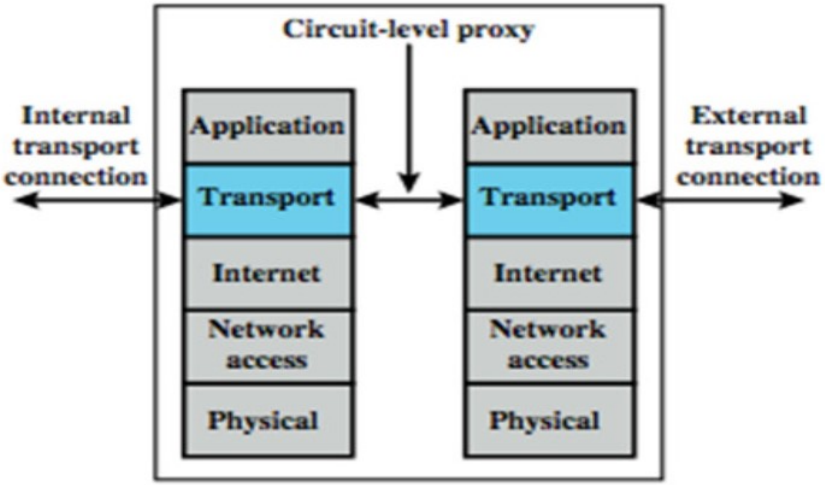
(A) Packet filtering firewall



(B) Stateful inspection firewall



(C) Application proxy firewall

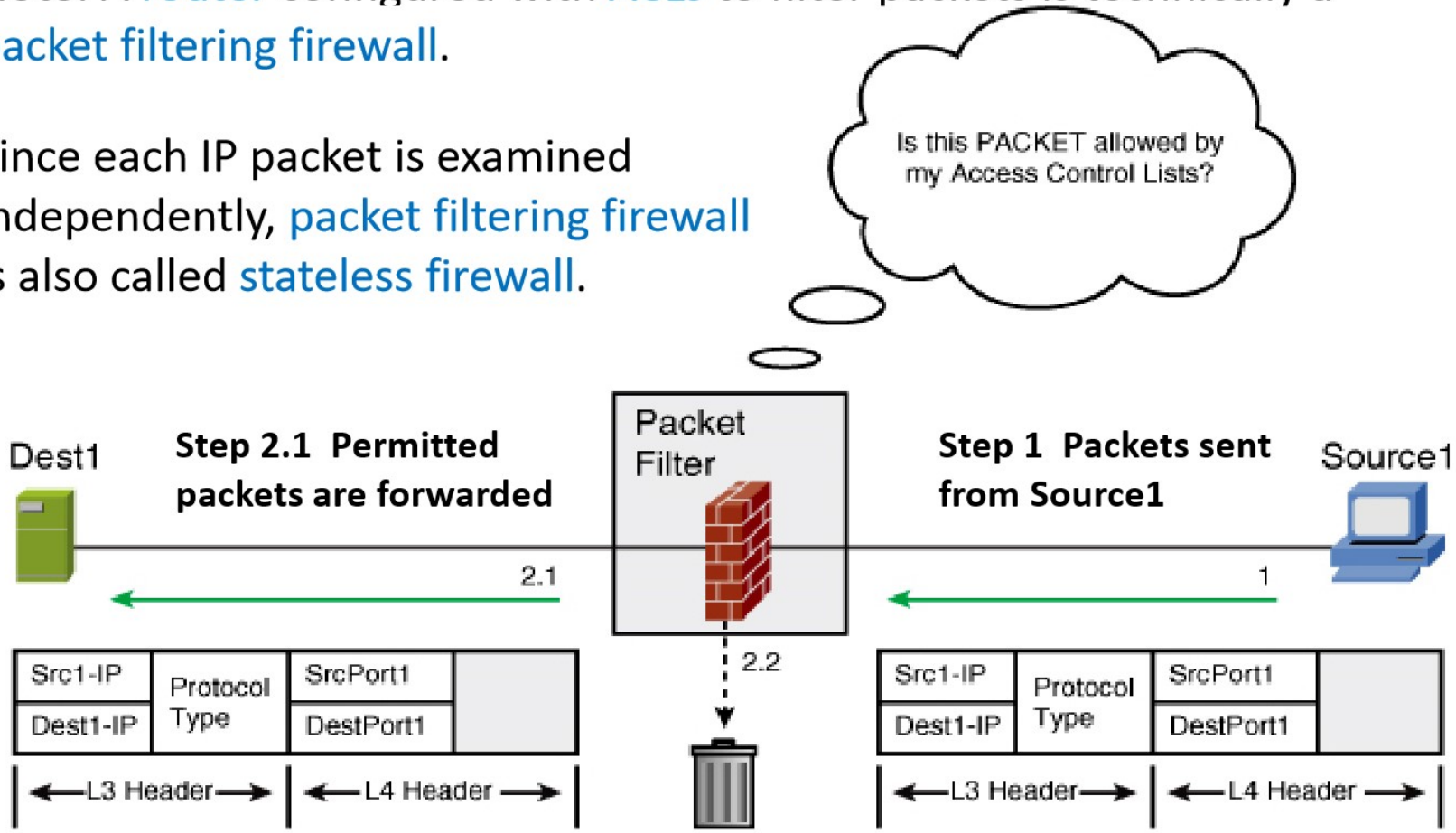


(D) Circuit-level proxy firewall

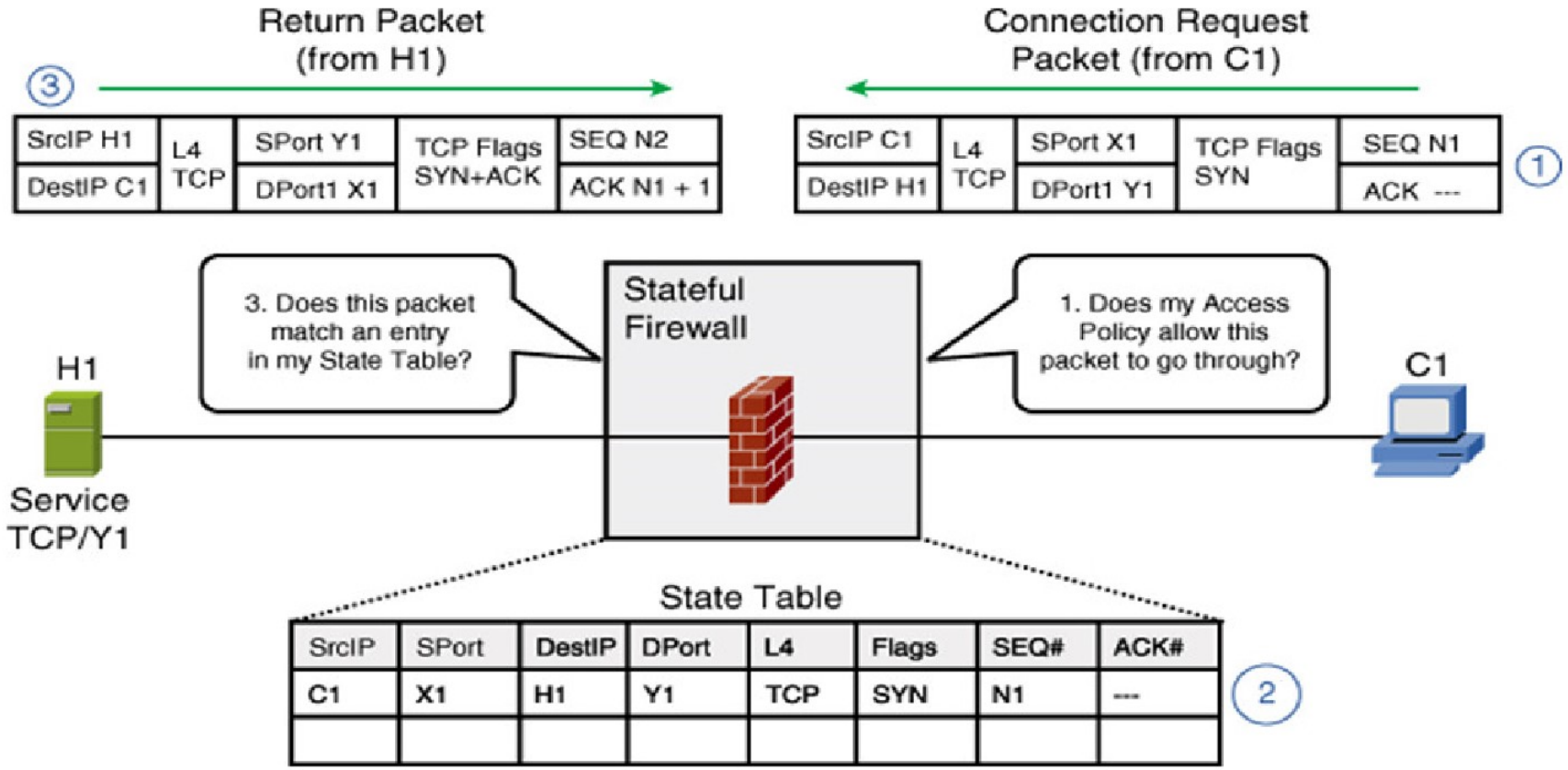
(A) In packet filtering firewall, access control policy rules are implemented as ACLs to filter each IP packet independently, but have limitations as discussed in previous lectures.

Note: A router configured with ACLs to filter packets is technically a packet filtering firewall.

Since each IP packet is examined independently, packet filtering firewall is also called stateless firewall.

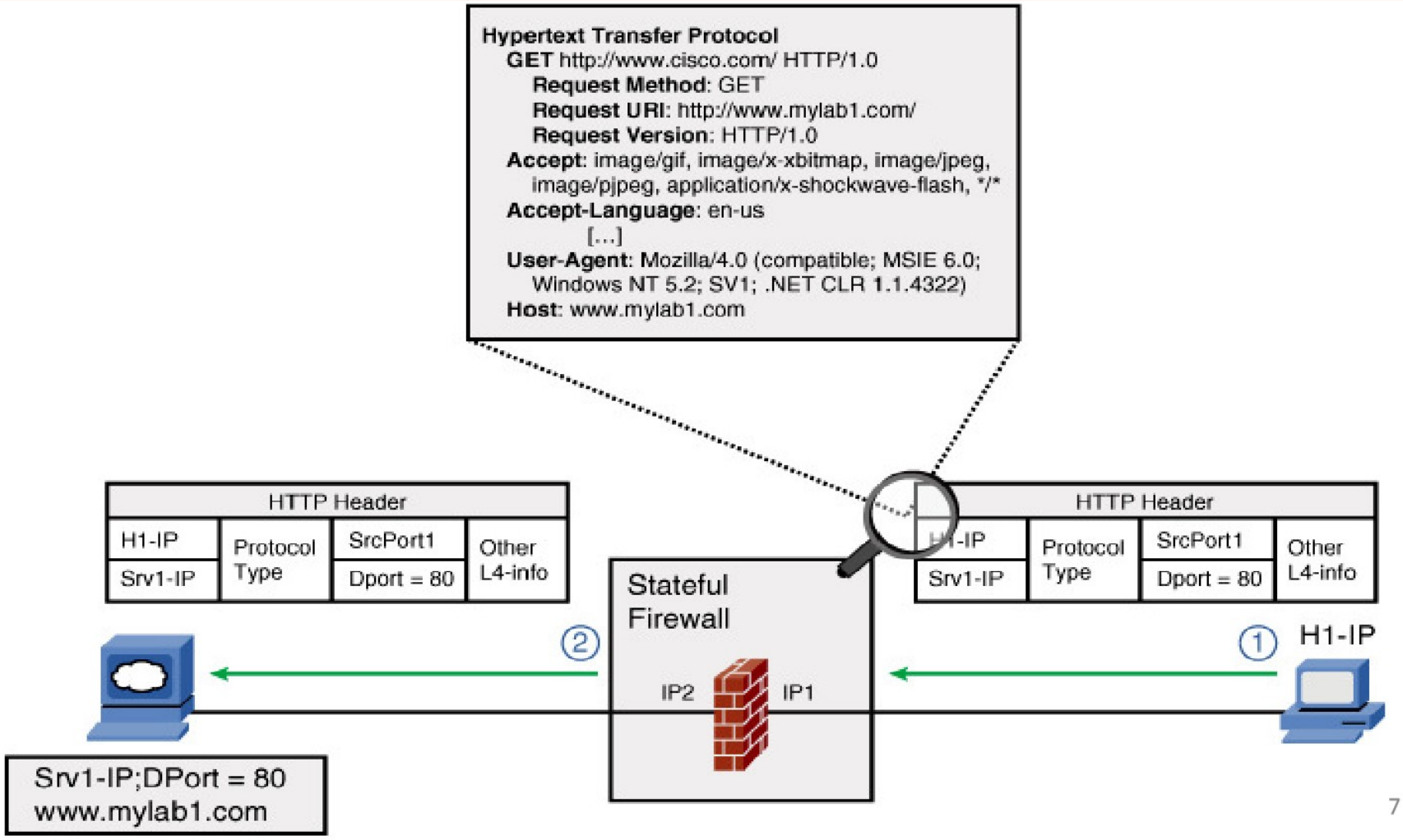


(B) To provide better security, stateful inspection firewalls are developed which maintain connection information in state table for packet filtering, and are commonly used today.

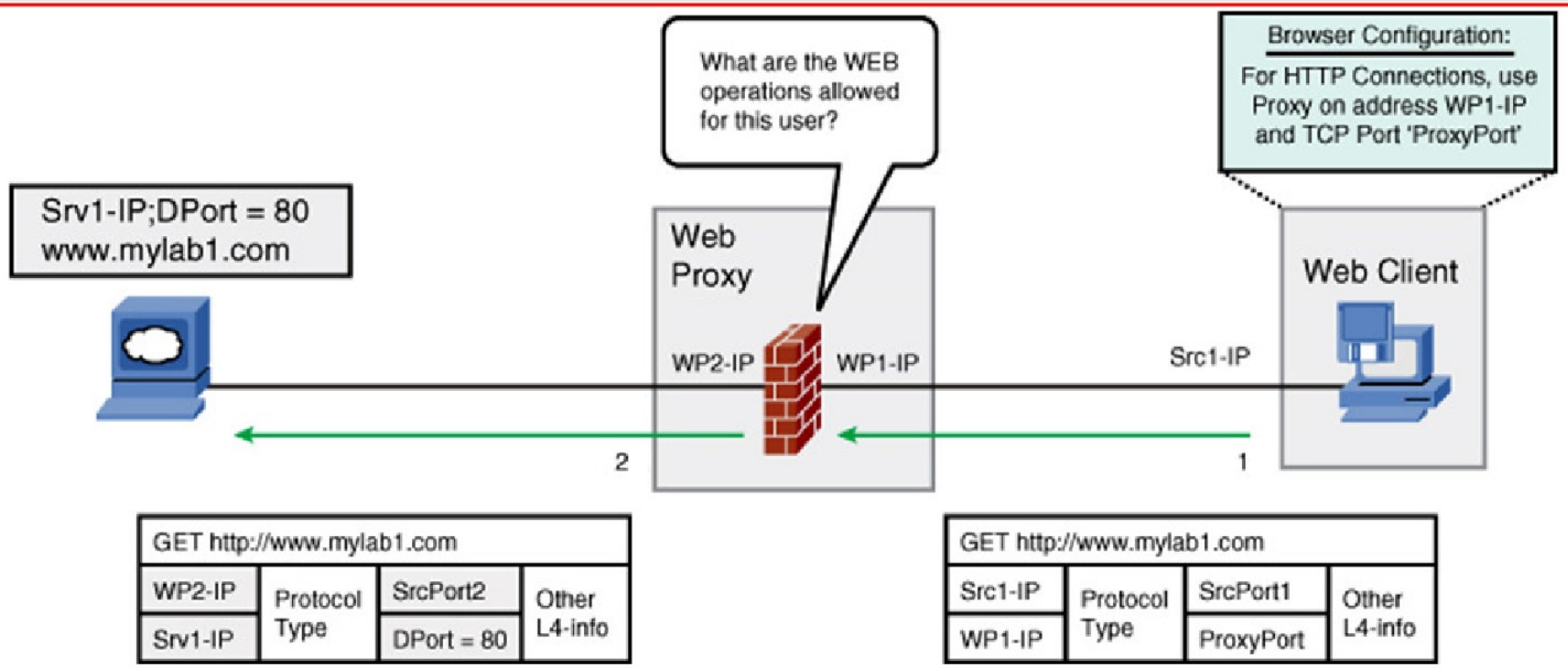


2 For permitted packet, an entry is created in firewall state table which will be used to verify return packets.

To achieve even better access control, modern stateful firewalls are often application-aware, i.e. able to inspect application-level protocol such as HTTP, e.g. Cisco ASA firewall.



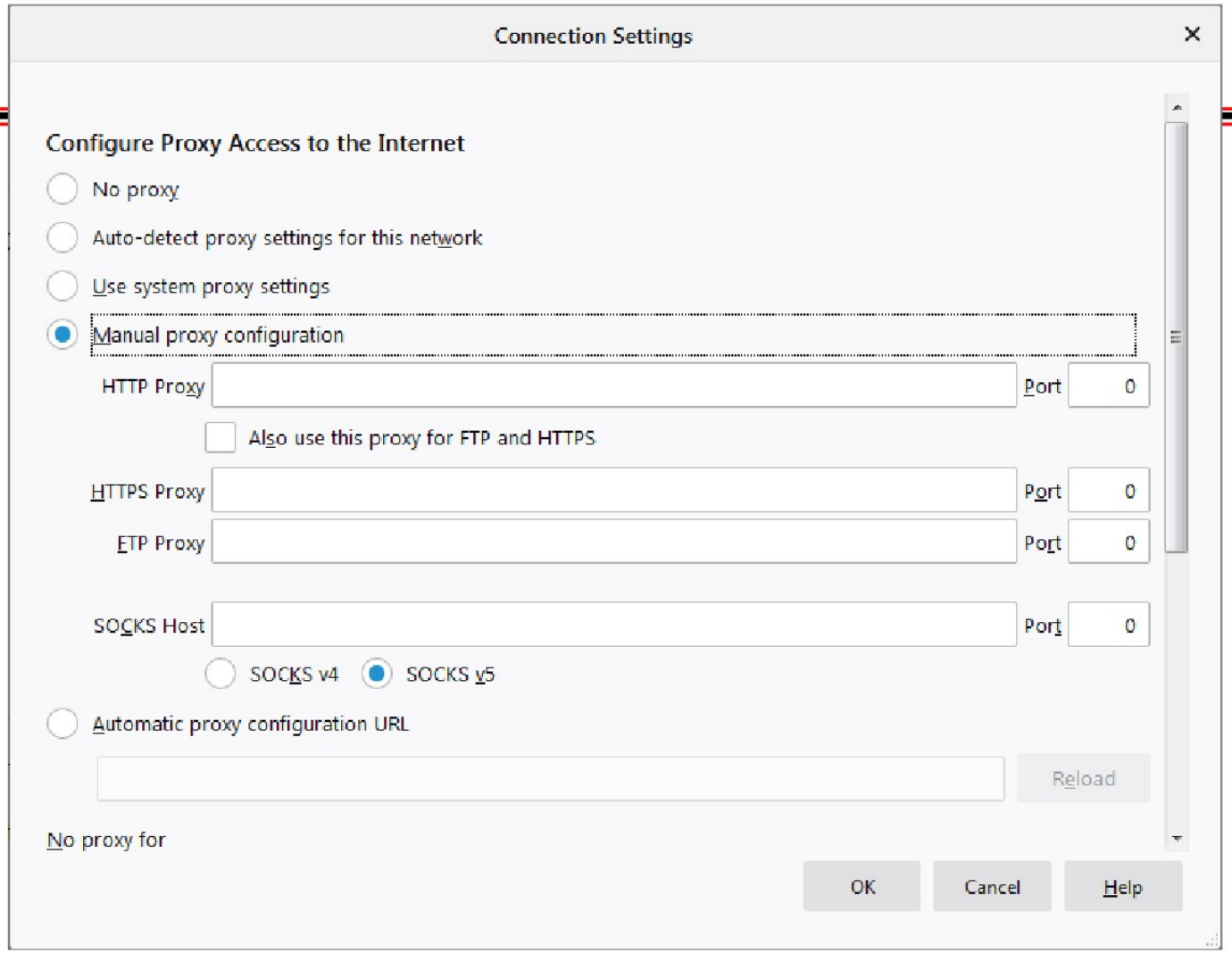
(C) Application proxy firewall, aka as application layer gateway (ALG) is specially designed to filter certain application, e.g. web proxy or WAF (Web Application Firewall).



Proxy firewall and application-aware stateful firewall are different:

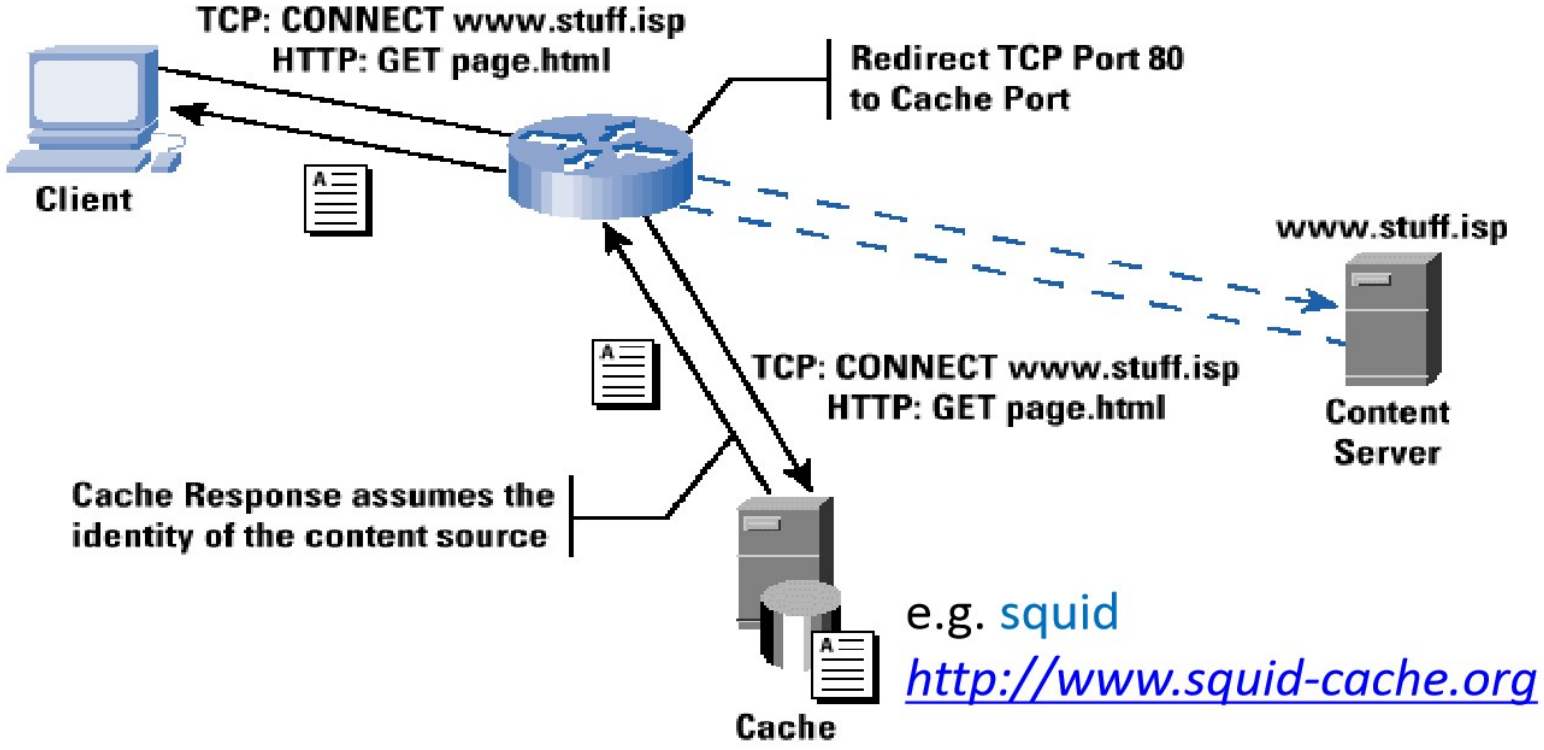
- Proxy firewall: two spliced connections between user and proxy, and proxy and destination server
- Stateful firewall: one direct connection between user and server

An example of configuring proxy in Firefox browser:



Nowadays, it is more common to have transparent proxy which will automatically intercept and restrict what users are allowed to access.

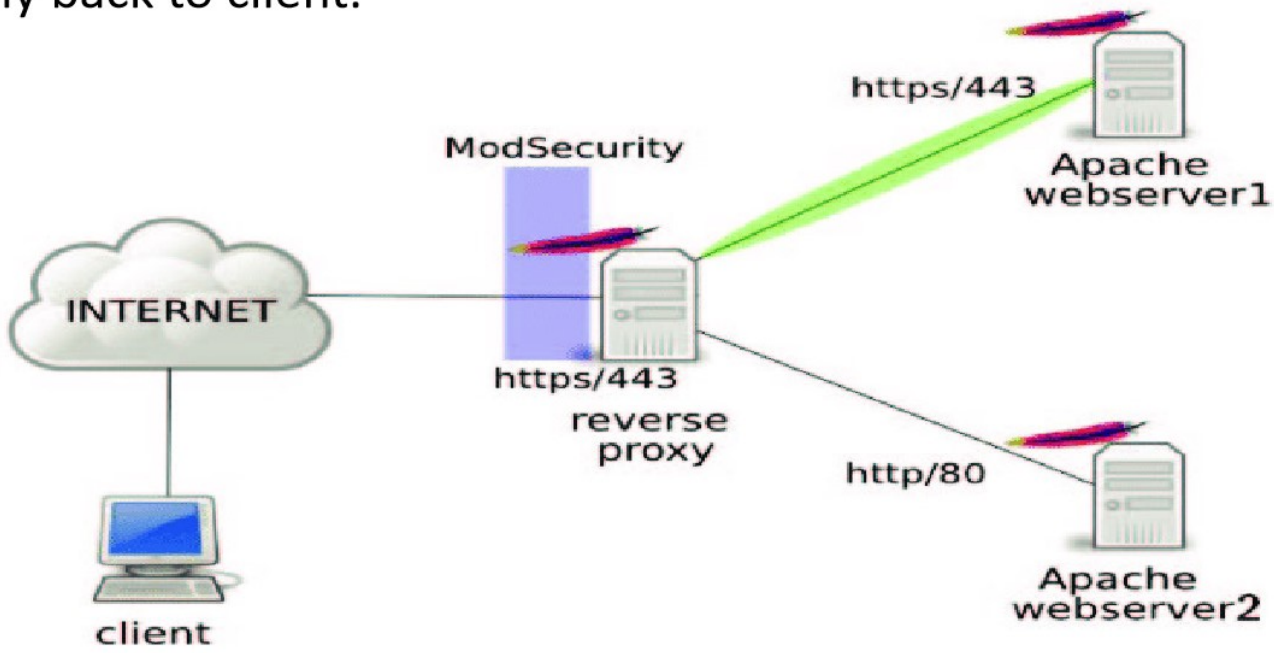
An example of transparent web proxy is the free open source Squid with WCCPv2 router interception which you may try in your team project.
https://wiki.squid-cache.org/Features/Wccp2



e.g. squid
http://www.squid-cache.org

Application proxy firewall may also be deployed in reverse proxy mode; e.g. WAFs to protect web servers.

In reverse proxy mode, client will connect to WAF which acts as the web server. If request is non-malicious, WAF will relay request to web server and reply back to client.



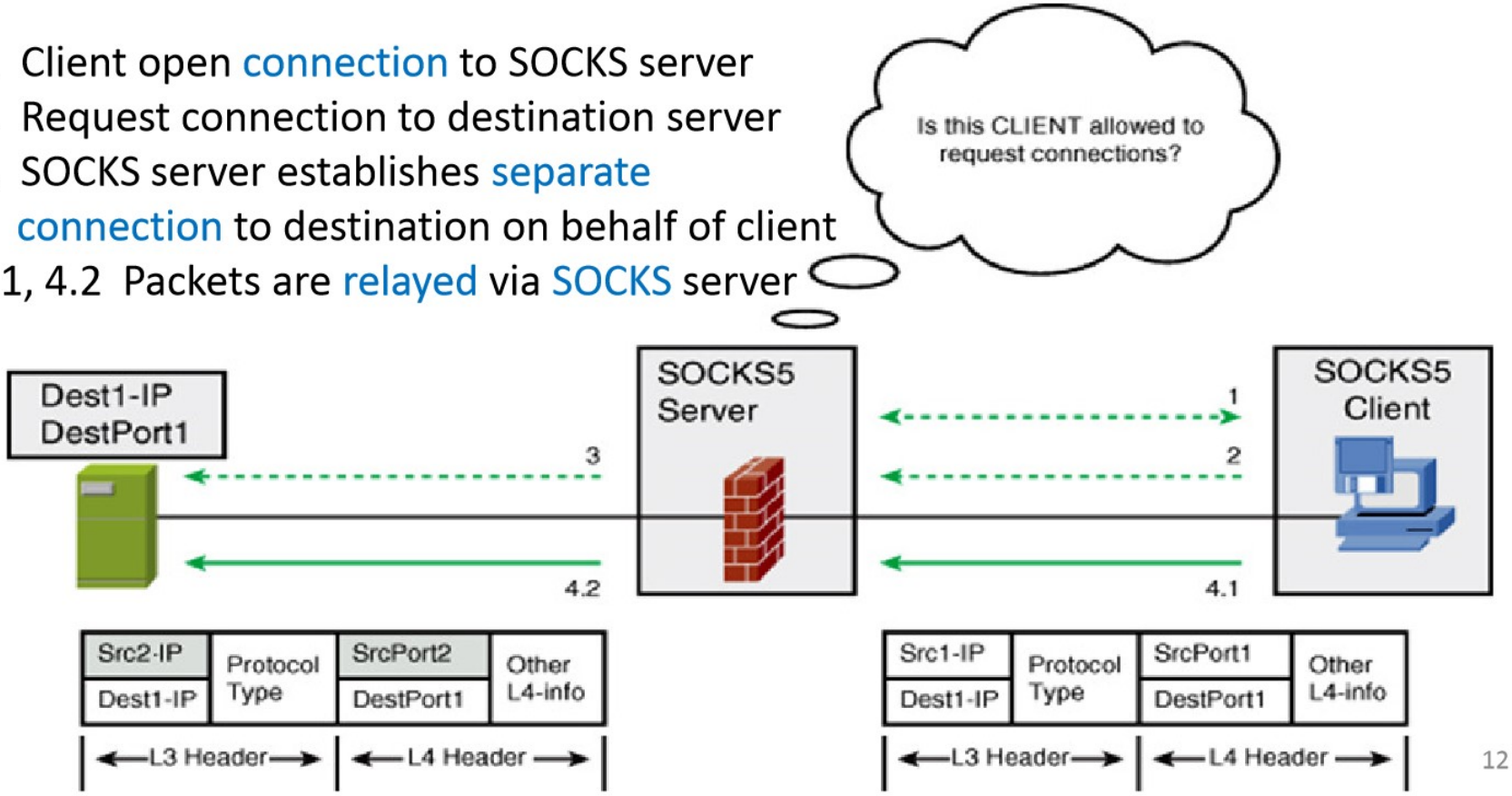
An example of WAF is the free open source ModSecurity which you may also try in your team project. https://github.com/SpiderLabs/ModSecurity

(D) Circuit-level proxy firewall, also referred as generic proxy works similarly by connecting to server on behalf of user, but does not understand application protocols.

Circuit-level proxy is commonly implemented using SOCKS5 (Socket Secure) protocol (RFC 1928), hence also called SOCKS server.

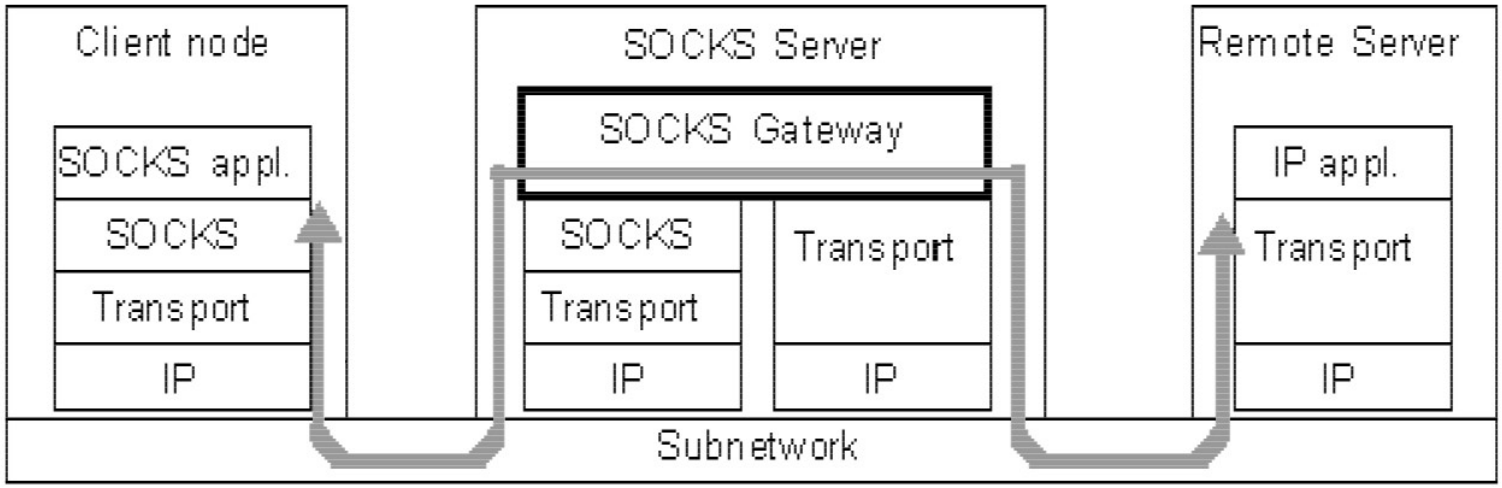
1. Client open connection to SOCKS server
2. Request connection to destination server
3. SOCKS server establishes separate connection to destination on behalf of client

4.1, 4.2 Packets are relayed via SOCKS server



Technically, SOCKS works as a 'shim-layer' between transport and application layers to carry any application-layer traffic.

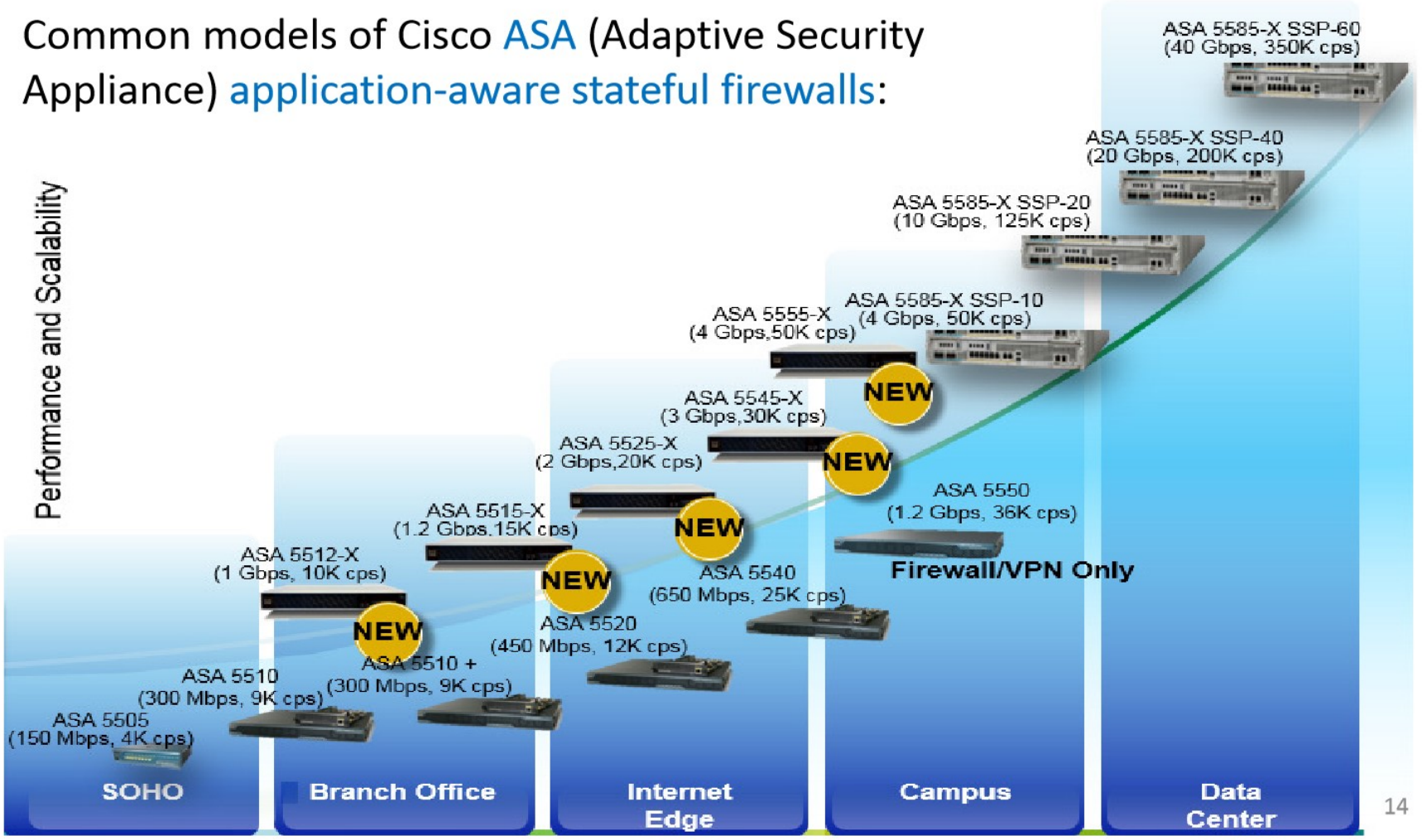
With circuit-level proxy, instead of opening socket to transport layer directly, applications are modified to call SOCKS5 API to carry its traffic to the firewall, hence called SOCKS applications.



The SOCKS server/firewall will then connect to destination server on behalf of users, and then relay the traffic between them.

To have a better understanding of firewall, we'll now study the Cisco ASA firewall which is an application-aware stateful firewall.

Common models of Cisco ASA (Adaptive Security Appliance) application-aware stateful firewalls:

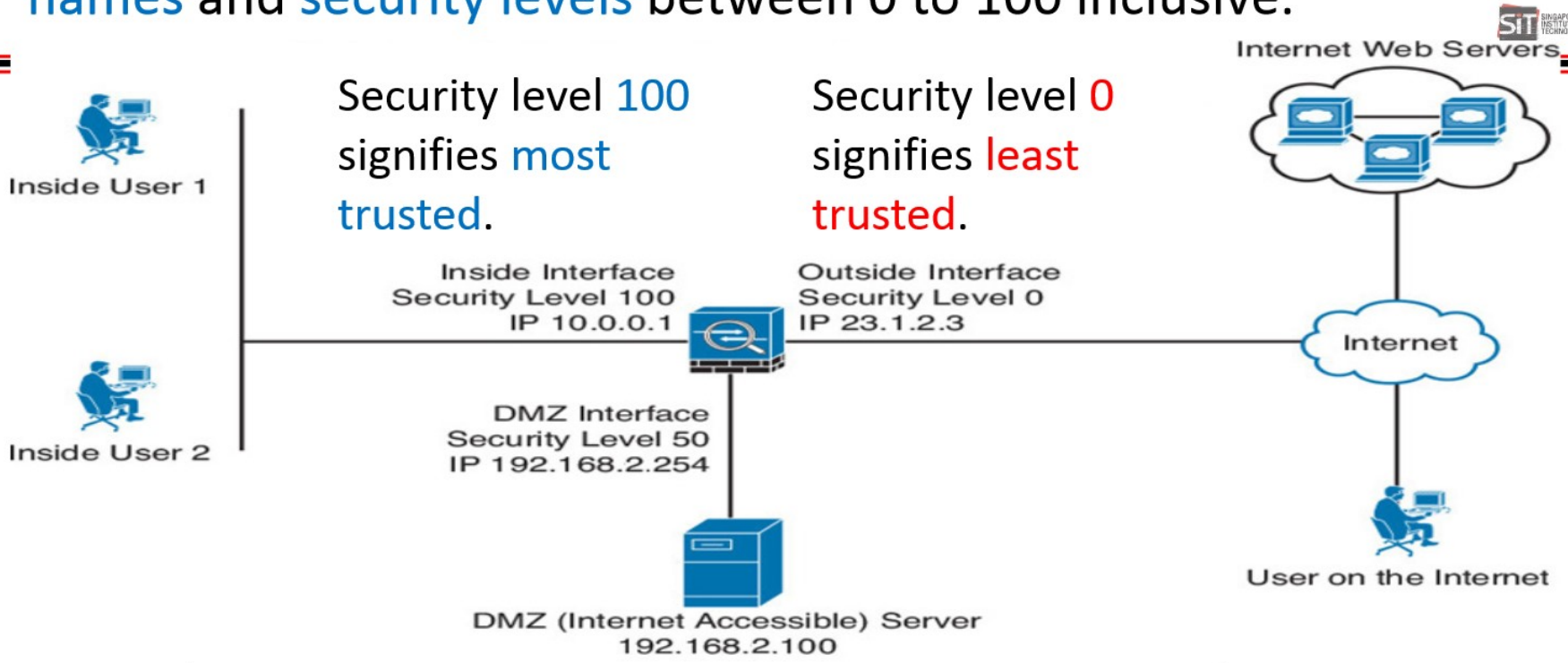


Lab Exercise 1.1

SINGAPORE INSTITUTE OF TECHNOLOGY

1.1 To get Cisco ASA up and operational, configure interface security level to segment networks into different security zones.

ASAs are typically deployed in routed mode like routers and its interfaces must be configured with IP addresses, and also names and security levels between 0 to 100 inclusive.




```
ciscoasa(config)# interface g0/1
ciscoasa(config-if)# nameif dmz
ciscoasa(config-if)# security-level 50
ciscoasa(config-if)# ip address 192.168.2.254 255.255.255.0
ciscoasa(config-if)# no shutdown
```

To verify that the interfaces of ASA are up and operational after configuration, use the similar CLI command as routers but note the slight difference as shown:

```
ciscoasa# show interface ip brief
Interface                  IP-Address      OK? Method Status                Protocol
GigabitEthernet0/0         23.1.2.3        YES manual up                    up
GigabitEthernet0/1         unassigned      YES unset  administratively down down
GigabitEthernet0/2         unassigned      YES unset  administratively down down
     .
     .
     .
Management0/0              192.168.1.1     YES manual up                    up
ciscoasa#
```

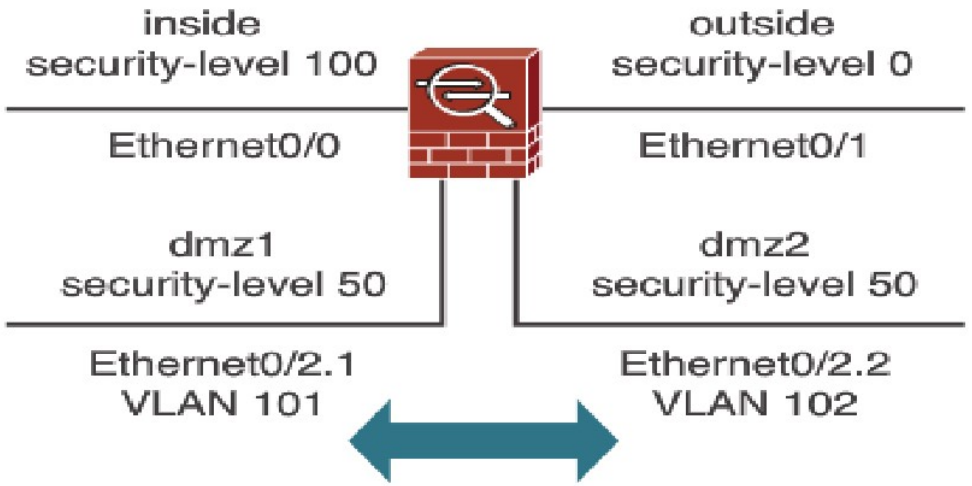
To show the details of each interface:

```
ciscoasa# show interface if_number
```

```
ciscoasa# show interface name
```

Note that if you choose to configure two or more interfaces with the same security level, traffic are by default not allowed to pass between them.

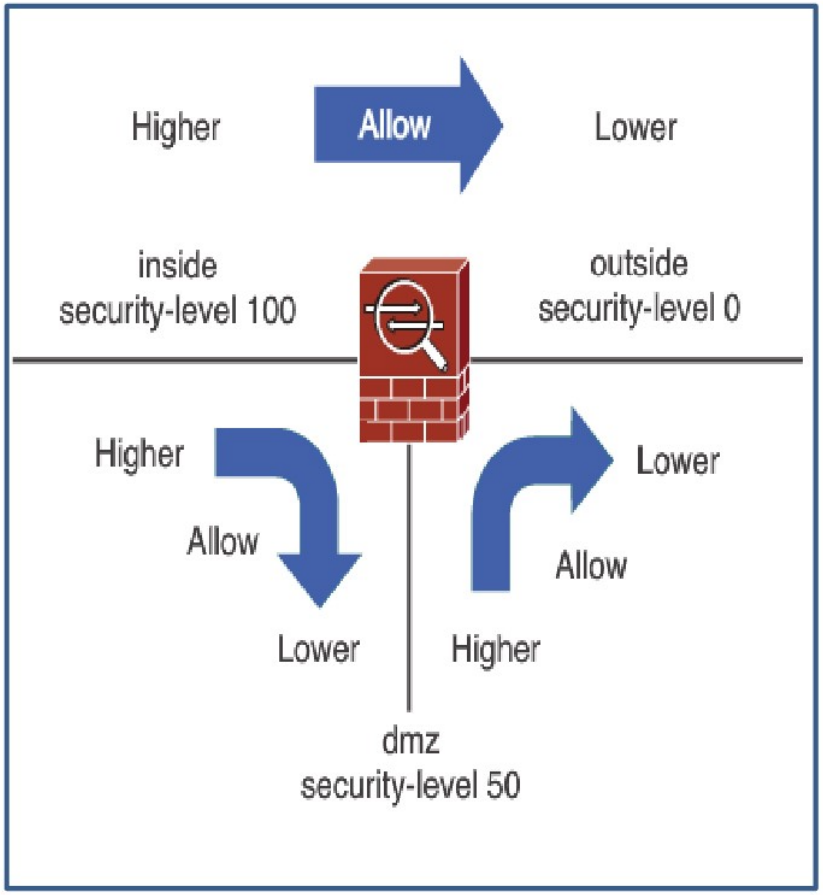
e.g.: you can use any security level between 0 and 100 to configure DMZ, but say you choose to use the same security level 50 for dmz1 and dmz2 as follows:

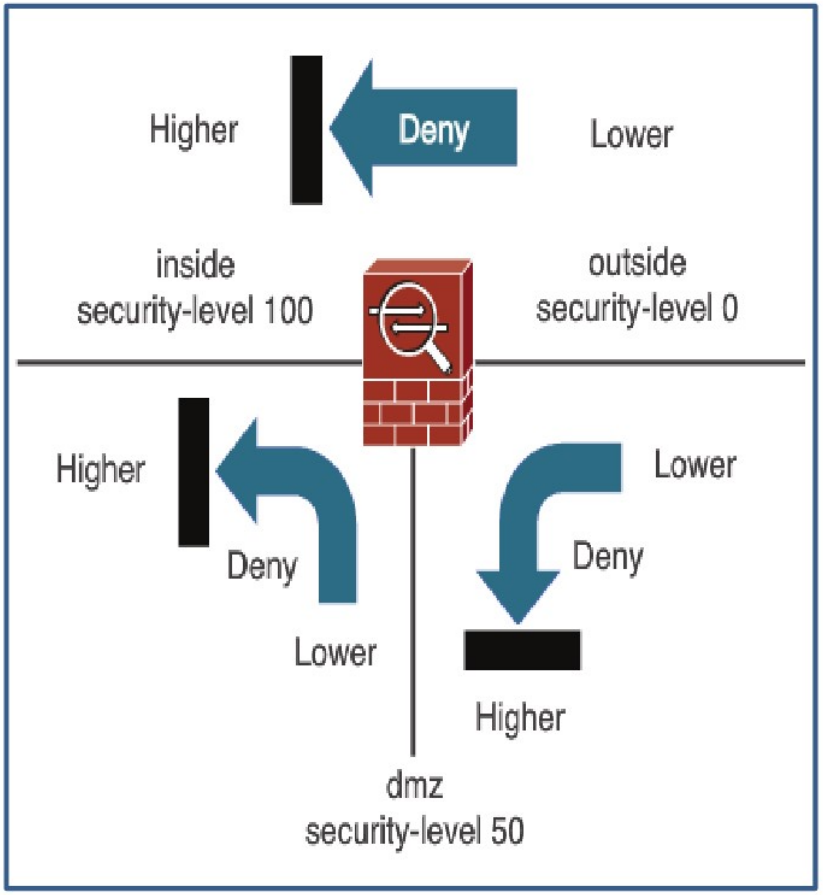


To override default security policy and allow traffic to be initiated from same security level:

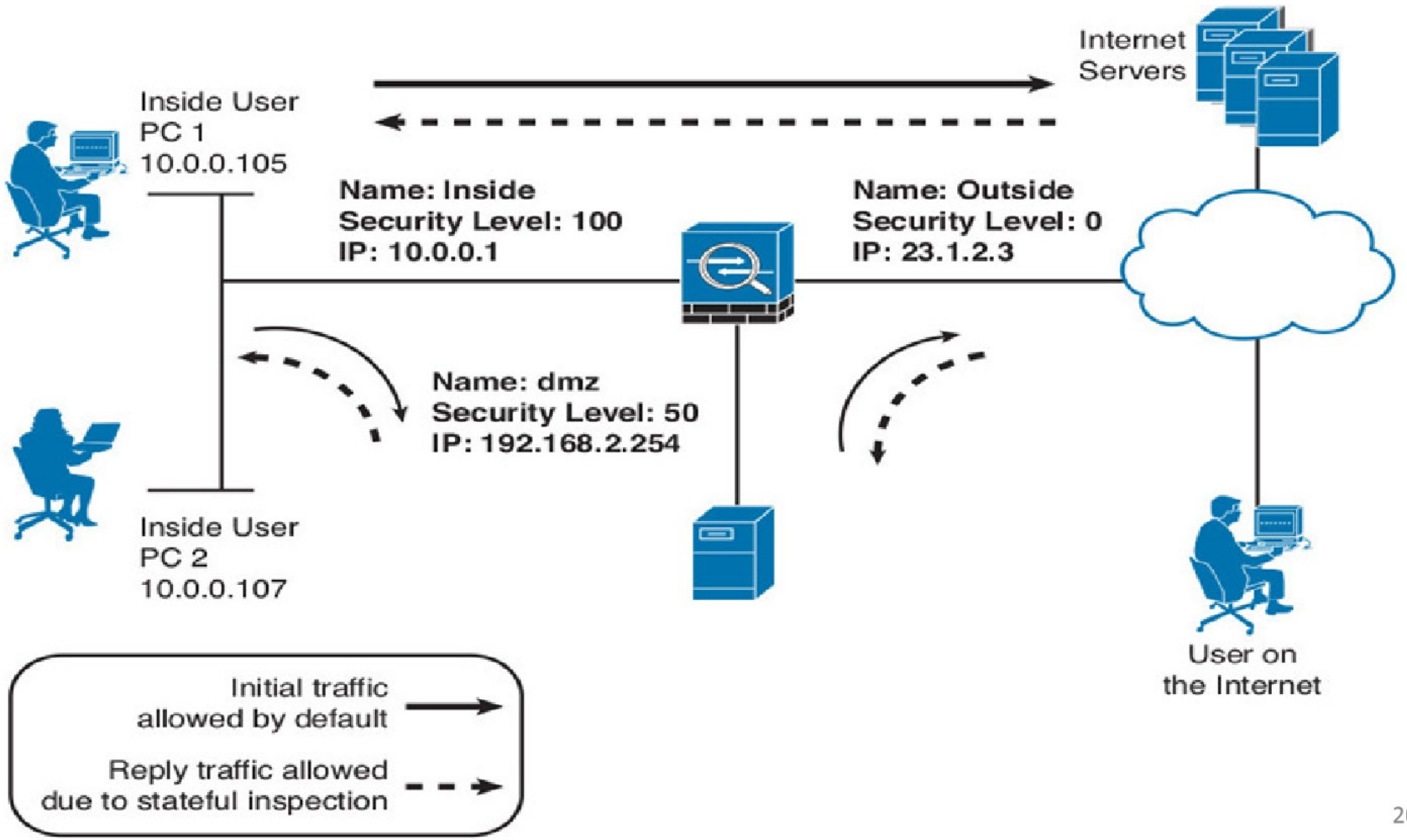
```
ciscoasa(config)# same-security-traffic permit inter-interface
```

By default, traffic initiated from higher security level to lower security level are allowed but not the reverse.





Nevertheless, due to stateful firewall feature which maintains state tables of initiated connections, return traffic from lower to higher security level are automatically allowed.



Specifically, one of the state tables maintained by ASA is the connection table which automatically tracks the connection parameters of TCP and UDP:

Protocol	Extent of Stateful Tracking
TCP	Source and destination addresses and ports, TCP flags, sequence numbers (in both directions), and idle time
UDP	Source and destination addresses and ports, idle time, and for some applications, request identifiers

To view the content of connection table:

```
ciscoasa# show conn
352 in use, 5985 most used
UDP DMZ 172.16.0.25:161 inside 10.0.0.30:1879, idle 0:00:27, bytes 706509, flags -
UDP outside 192.0.2.18:123 inside 10.0.0.108:123, idle 0:00:34, bytes 79, flags -
TCP outside 192.0.2.150:80 DMZ 172.16.0.5:59512, idle 0:00:00, bytes 0, flags Uf
TCP outside 192.0.2.150:80 inside 10.0.0.102:59559, idle 0:00:03, bytes 1488,
   flags UfFRIO
...
```

Or to view more details:

```
ciscoasa# show conn detail
```

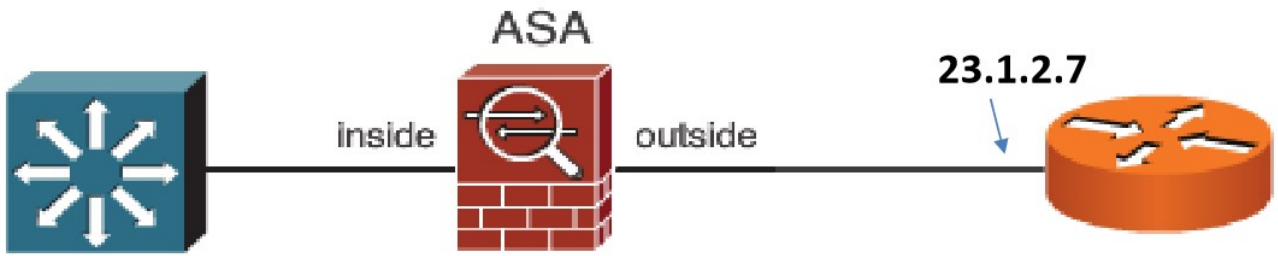
Another state table in ASA is the local host table which maintains per-host statistics as follows:

```
FIREWALL# show local-host 10.0.0.108
Interface management: 0 active, 0 maximum active, 0 denied
Interface DMZ: 2 active, 69 maximum active, 0 denied
Interface inside: 130 active, 320 maximum active, 0 denied
local host: <10.0.0.108>,
    TCP flow count/limit = 15/unlimited
    TCP embryonic count to host = 0
    TCP intercept watermark = unlimited
    UDP flow count/limit = 10/unlimited

  Xlate:
    NAT from any:10.0.0.108 to outside:209.165.200.238 flags I idle 0:00:00
      timeout 3:00:00

  Conn:
    TCP outside 192.0.2.65:22 inside 10.0.0.108:50124, idle 0:00:58, bytes 443,
      flags UIO
    ... output omitted ...
Interface outside: 120 active, 1940 maximum active, 0 denied
```

Note that for connection with other network devices, ASAs will also need to be configured with static and/or default routes, or routing protocols to learn routes dynamically.

To configure static route in ASA:

```
ciscoasa(config)# route interface ip_address netmask gateway_ip
```



e.g.: Suppose your ISP can be reached via 23.1.2.7, you can configure a static default route in ASA as follows:

```
ciscoasa(config)# route outside 0.0.0.0 0.0.0.0 23.1.2.7
```

To verify the routing table in the ASA, use the similar command as in routers but note the slight difference as shown:

```
ciscoasa# show route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 23.1.2.7 to network 0.0.0.0

C     23.1.2.0 255.255.255.0 is directly connected, outside
C     192.168.1.0 255.255.255.0 is directly connected, management
S*    0.0.0.0 0.0.0.0 [1/0] via 23.1.2.7, outside
ciscoasa#
```

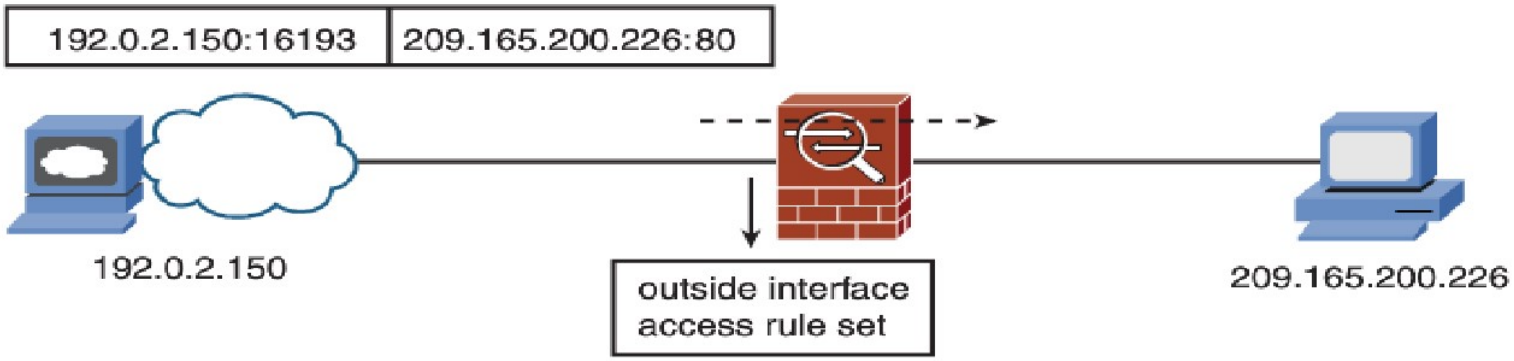
Lab Exercise 1.2

SINGAPORE INSTITUTE OF TECHNOLOGY

1.2 Configure access control list in Cisco ASA to override security level if needed.

If needed, ACLs can be configured at ASA to override security level; e.g. to allow lower security-level to access higher security-level, or limit what higher-security level can send.

e.g.: To allow outside users to access web server 209.165.200.226 at DMZ




```
ciscoasa(config)# access-list OUTSIDE_TO_DMZ permit tcp any
host 209.165.200.226 eq www
ciscoasa(config)# access-group OUTSIDE_TO_DMZ in interface
outside
```

Note some differences between configuring ACLs in ASAs and routers:

1. ASAs assume ACLs are extended and 'extended' keyword is optional
2. ASAs use standard subnet mask but not wildcard mask to specify IP address block

Similar as Cisco IOS routers, line numbers are automatically added into ACLs in ASAs to support editing as follows:

To show ACL, use the command:

```
ciscoasa# show access-list OUTSIDE_TO_DMZ
access-list OUTSIDE_TO_DMZ; 1 elements; name hash: 0x5113f0a2
access-list OUTSIDE_TO_DMZ line 1 extended permit tcp any host
209.165.200.226 eq www (hitcnt=9) 0x4a02e414
```

To insert an ACL statement, use 'line' keyword to indicate the position:

```
ciscoasa(config)# access-list OUTSIDE_TO_DMZ line 1 remark
allow internet access to web
```

To delete an ACL statement, use 'no' keyword in front:

```
ciscoasa(config)# no access-list OUTSIDE_TO_DMZ line 1 remark
allow internet access to web
```

To delete the whole ACL:

```
ciscoasa(config)# clear configure access-list OUTSIDE_TO_DMZ
```

Lab Exercise 1.3

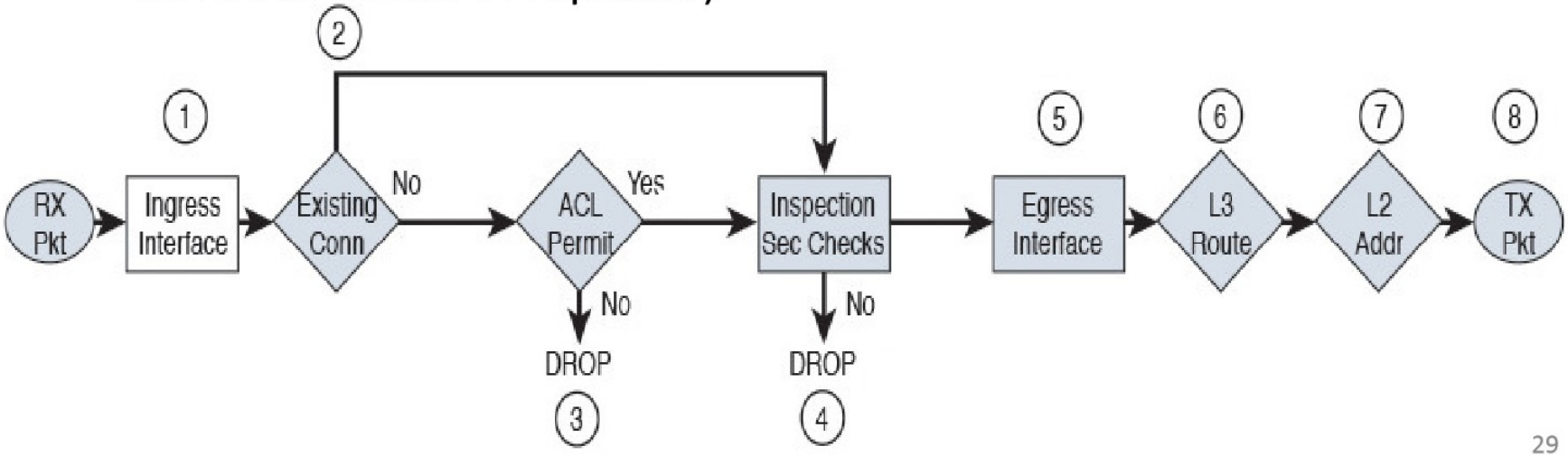
SINGAPORE INSTITUTE OF TECHNOLOGY

1.3 Configure ASA service policy using Modular Policy Framework (MPF) for advanced features, e.g. enabling inspection engine, etc.

Before discussing the configuration of Modular Policy Framework (MPF), it's useful to understand packet processing order of operations in ASA.

When a packet arrives at ASA and before it is allowed to pass through, it will go through the following filtering as shown:

2. States table (maintain automatically after packets are permitted by interface security level)
3. Access control list (to override interface security level if needed)
4. MPF inspection security policy (shipped with a factory default, but can be modified or replaced)

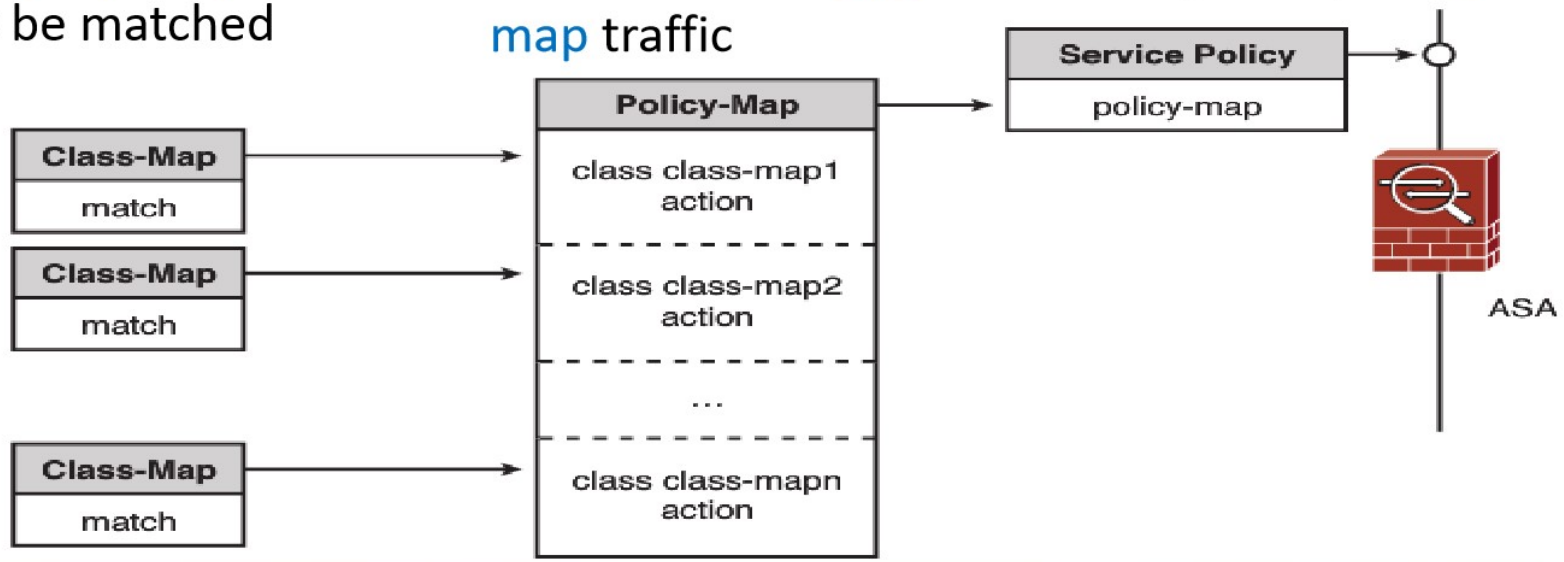


Essentially, configuring ASA using Modular Policy Framework (MPF) involves 3 steps:

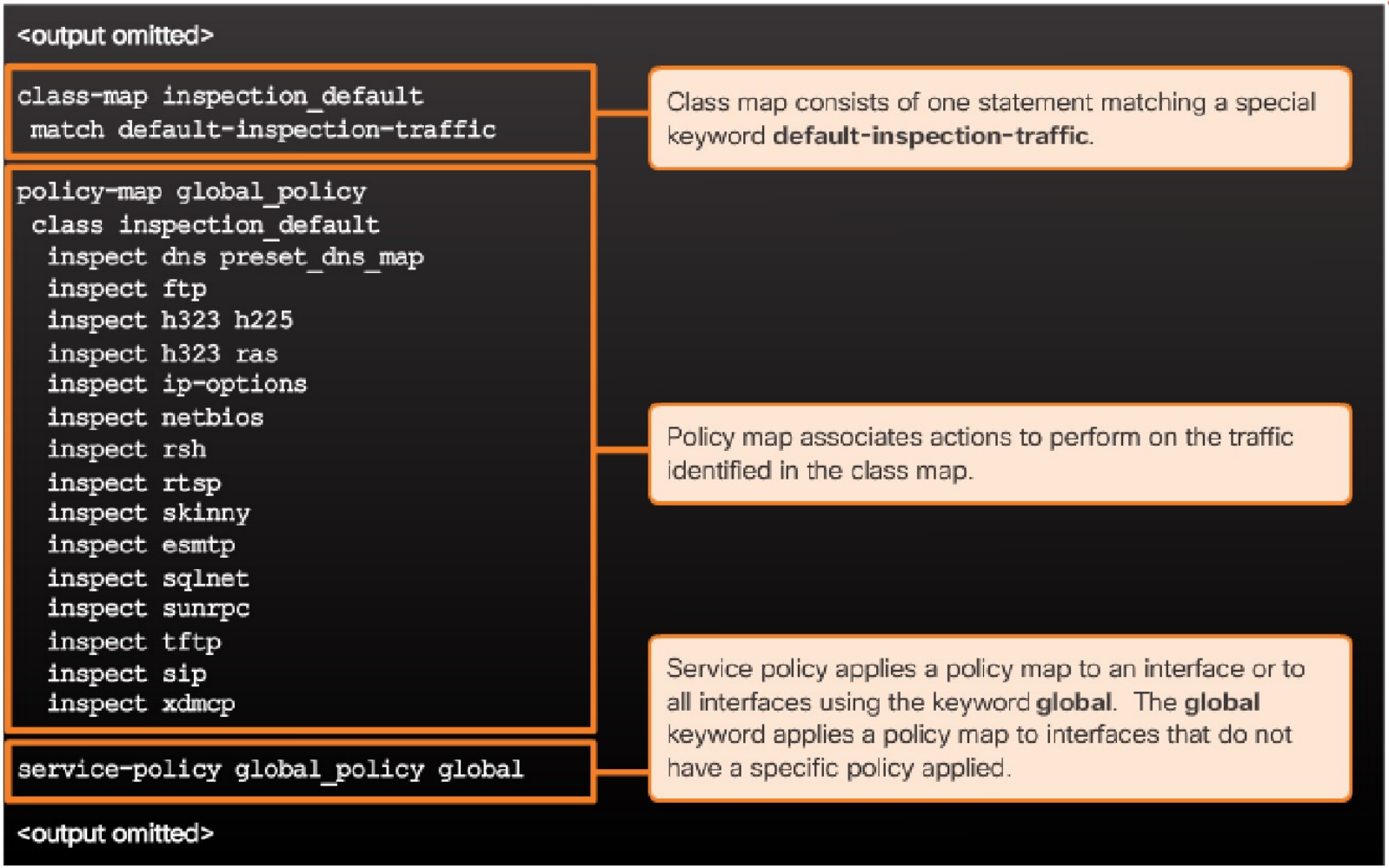
1. Configure class-map: specify traffic to be matched

2. Configure policy-map: specify actions to be taken on class-map traffic

3. Configure service-policy: apply policy-map to an interface or globally to all



By default, ASA is pre-configured with a class-map named inspection-default and a policy-map named global_policy which is applied globally using service-policy on all ASA interfaces.



Specifically, the default-inspection-traffic in class-map inspection_default is matching commonly encountered network traffic as follows:

```
default-inspection-traffic  Match default inspection traffic:

        ctiqbe----tcp--2748        dns-------udp--53
        ftp-------tcp--21          gtp-------udp--2123,3386
        h323-h225-tcp--1720        h323-ras--udp--1718-1719
        http------tcp--80          icmp------icmp
        ils-------tcp--389         ip-options-----rsvp
        mgcp------udp--2427,2727   netbios---udp--137-138
        radius-acct---udp--1646    rpc-------udp--111
        rsh-------tcp--514         rtsp------tcp--554
        sip-------tcp--5060        sip-------udp--5060
        skinny----tcp--2000        smtp------tcp--25
        sqlnet----tcp--1521        tftp------udp--69
        waas------tcp--1-65535     xdmcp-----udp--177
```

To view statistics of the matching traffic inspected by the ASA, use the show command as follows:

```
ciscoasa# show service-policy
Global policy:
  Service-policy: global_policy
    Class-map: inspection_default
      Inspect: dns preset_dns_map, packet 0, drop 0, reset-drop 0
      Inspect: ftp, packet 0, drop 0, reset-drop 0
      Inspect: h323 h225 _default_h323_map, packet 0, drop 0, reset-drop 0
               tcp-proxy: bytes in buffer 0, bytes dropped 0
      Inspect: h323 ras _default_h323_map, packet 0, drop 0, reset-drop 0
      Inspect: rsh, packet 0, drop 0, reset-drop 0
      Inspect: rtsp, packet 0, drop 0, reset-drop 0
               tcp-proxy: bytes in buffer 0, bytes dropped 0
      Inspect: esmtp _default_esmtp_map, packet 0, drop 0, reset-drop 0
      Inspect: sqlnet, packet 0, drop 0, reset-drop 0
      Inspect: skinny, packet  0, drop 0, reset-drop 0
               tcp-proxy: bytes in buffer 0, bytes dropped 0
      Inspect: sunrpc, packet 0, drop 0, reset-drop 0
               tcp-proxy: bytes in buffer 0, bytes dropped 0
      Inspect: xdmcp, packet 0, drop 0, reset-drop 0
      Inspect: sip, packet 0, drop 0, reset-drop 0
               tcp-proxy: bytes in buffer 0, bytes dropped 0
      Inspect: netbios, packet 0, drop 0, reset-drop 0
      Inspect: tftp, packet 0, drop 0, reset-drop 0
      Inspect: ip-options _default_ip_options_map, packet 0, drop 0, reset-drop 0
```

To implement your required access control policy in ASA, you may modify the default policy-map as follows or configure your own MPF as discussed in the next few slides.

E.g.: For security reason, ICMP traffic is denied in ASA by default. But if your company's security policy allows ICMP, you can configure ASA to maintain stateful tracking of ICMP to permit return ICMP traffic by adding ICMP inspection engine in the default policy-map as shown below:

```
ciscoasa(config)# policy-map global_policy
ciscoasa(config-pmap)# class inspection_default
ciscoasa(config-pmap-c)# inspect icmp
ciscoasa(config-pmap-c)# exit
ciscoasa(config-pmap)# exit
ciscoasa(config)#
```

Note: You may also write your own policy-map and apply it on appropriate interface of the ASA.

Alternatively, to configure new MPF – Step 1: Configure MPF class map to specify what type of network traffic to be matched according to some conditions, e.g.:

```
ciscoasa(config)# class-map class_map_name
ciscoasa(config-cmap)# description text
ciscoasa(config-cmap)# match ...
ciscoasa(config-cmap)# exit
```

Common match commands for specifying traffic:

Matching Condition	Command Syntax
Any traffic	ciscoasa(config-cmap)# **match any**
Default traffic types	ciscoasa(config-cmap)# **match default-inspection-traffic**
Destination port number	ciscoasa(config-cmap)# **match port** {**tcp** | **udp**} {**eq** *port* | **range** *start end*}
Access list	ciscoasa(config-cmap)# **match access-list** *acl_name*
RTP port number range	ciscoasa(config-cmap)# **match rtp** *starting_port range*

An example of configuring MPF class map:

```
ciscoasa(config)# class-map anything
ciscoasa(config-cmap)# match any
ciscoasa(config-cmap)# exit
!
ciscoasa(config)# class-map voice
ciscoasa(config-cmap)# match rtp 2000 100
ciscoasa(config-cmap)# exit
!
ciscoasa(config)# access-list dc extended permit ip any 10.100.0.0 255.255.0.0
ciscoasa(config)# class-map data-center
ciscoasa(config-cmap)# match access-list dc
ciscoasa(config-cmap)# exit
```

Step 2: Configure MPF layer 3/4 policy map to specify one or more actions to take on the matched traffic, e.g.:

```
ciscoasa(config)# policy-map policy_map_name
ciscoasa(config-pmap)# description text
ciscoasa(config-pmap)# class class_map_name
ciscoasa(config-pmap-c)# ...
```

Some commands for configuring actions to take on matched traffic:

Action	Command Syntax
Set connection limits	ciscoasa(config-pmap-c)# **set connection** ...
Inspect applications	ciscoasa(config-pmap-c)# **inspect** *engine_name*
Police the traffic	ciscoasa(config-pmap-c)# **police** {**output** | **input**} *conform_rate* [*burst_bytes*] [**conform-action** {**drop** | **transmit**}] [**exceed-action** {**drop** | **transmit**}]
Apply priority handling	ciscoasa(config-pmap-c)# **priority**

An example of configuring MPF layer 3/4 policy map based on previously configured class maps:

```
ciscoasa(config)# policy-map p1
ciscoasa(config-pmap)# class anything
ciscoasa(config-pmap-c)# set connection ...
ciscoasa(config-pmap-c)# inspect ...
ciscoasa(config-pmap-c)# exit
ciscoasa(config-pmap)# class voice
ciscoasa(config-pmap-c)# priority
ciscoasa(config-pmap-c)# exit
ciscoasa(config-pmap)# class data-center
ciscoasa(config-pmap-c)# set connection timeout ...
ciscoasa(config-pmap)# exit
```

Note:

- When a packet matches a class map for a feature type, ASA does not match it to any subsequent class maps for same feature type.
- If the packet matches a subsequent class map for a different feature type, then ASA also applies the actions for the subsequent class map.

For more information, refer
https://www.cisco.com/c/en/us/td/docs/security/asa/asa96/configuration/firewall/asa-96-firewall-config/inspect-service-policy.html

Step 3: Configure MPF service policy to apply policy map to the ASA interface or globally as follows:

```
ciscoasa(config)# service-policy policy_map_name {global | interface if_name}
```

Based on where service policy is configured, the actions for different feature types will be applied to traffic bi-directionally or uni-directionally:

Feature	Single Interface Direction	Global Direction
Application inspection (multiple types)	Bidirectional	Ingress
NetFlow Secure Event Logging filtering	N/A	Ingress
QoS input policing	Ingress	Ingress
QoS output policing	Egress	Egress
QoS standard priority queue	Egress	Egress
TCP and UDP connection limits and timeouts, and TCP sequence number randomization	Bidirectional	Ingress
TCP normalization	Bidirectional	Ingress
TCP state bypass	Bidirectional	Ingress

- Each interface can have only one service-policy applied to it. In addition, only one global policy can be applied to all interfaces.
- Interface service-policy takes precedence over global service-policy.

Lab Exercise 2

SINGAPORE INSTITUTE OF TECHNOLOGY

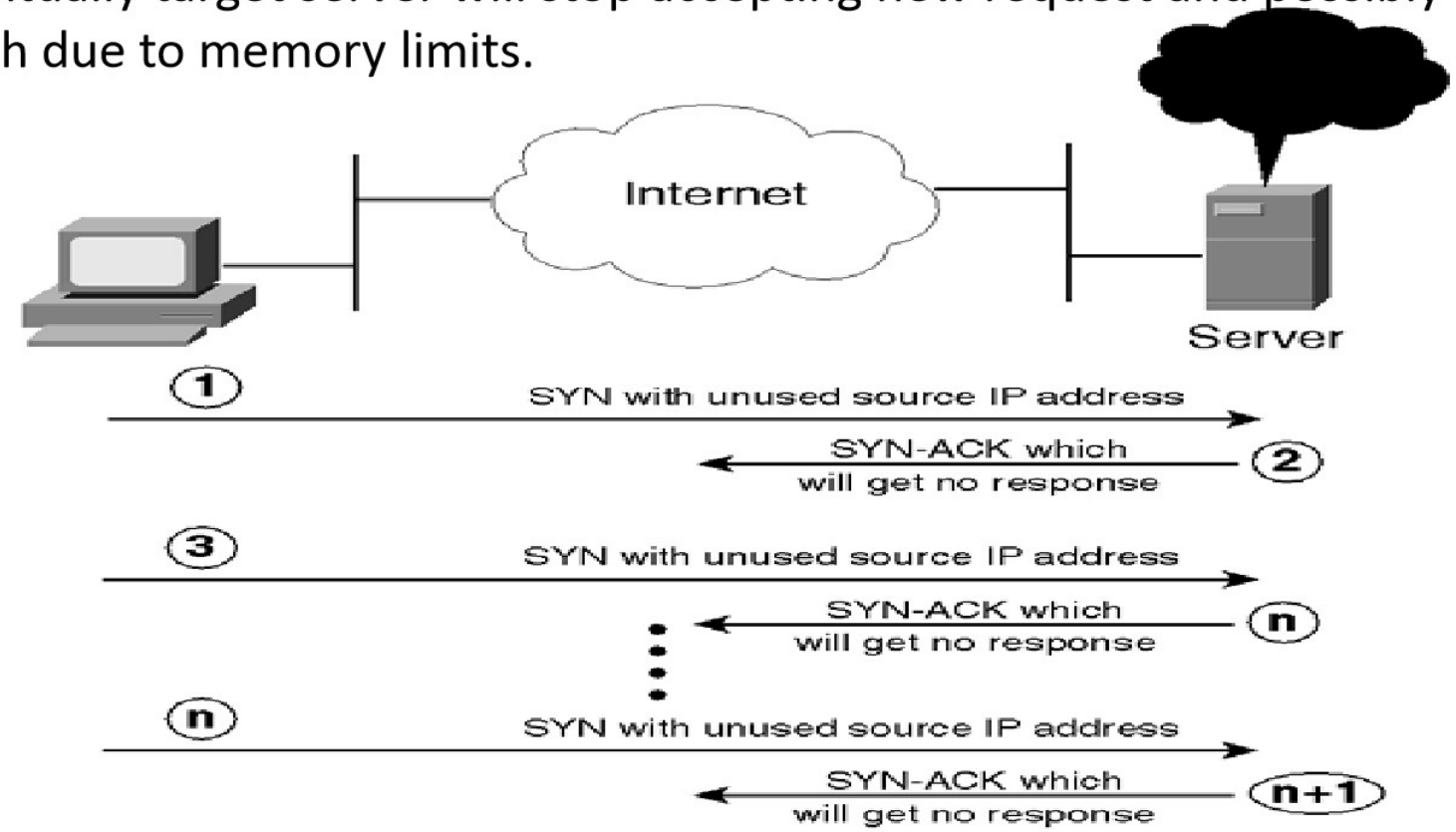
2.1 TCP SYN flooding attack

2.2 Defense:

- Configure MPF service policy in ASA

TCP SYN flooding attack is an effective DoS attack which sends TCP SYN segments to a target server but deliberately never completes the TCP handshake.

(Recall ICT1010 on TCP 3-way handshake.) Sending SYN request without subsequent ACK will result in half-open (embryonic) connection. Eventually target server will stop accepting new request and possibly crash due to memory limits.

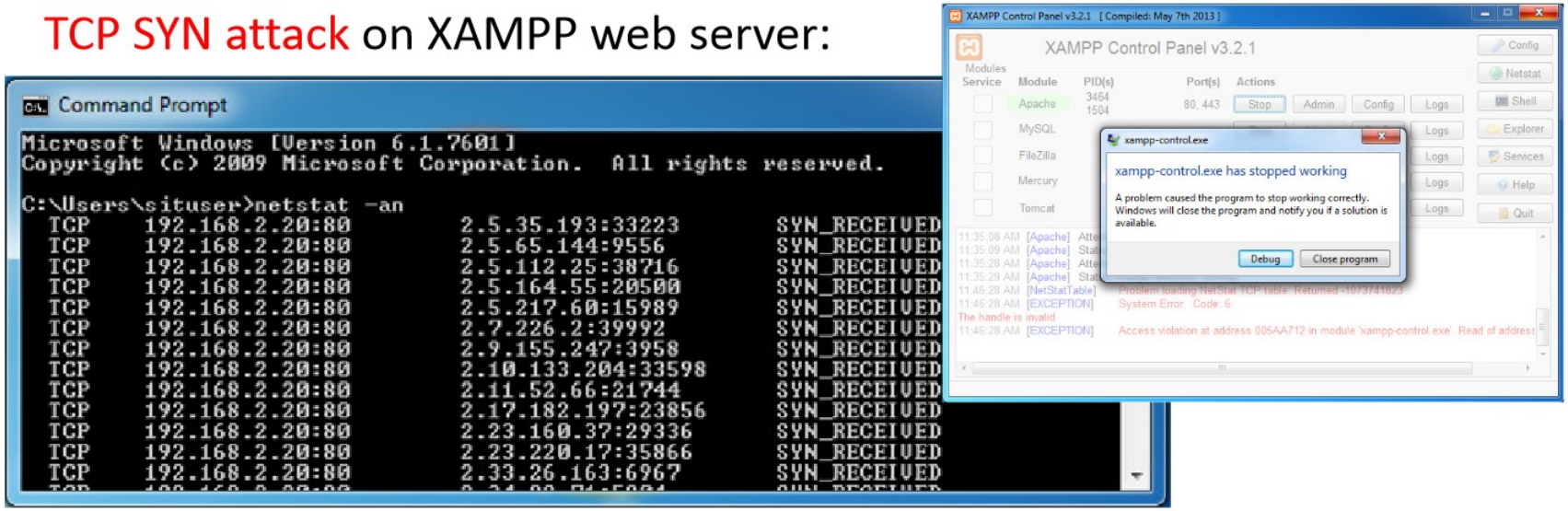


hping3 in Kali Linux may be used to conduct TCP SYN attack which you will try in the lab.

e.g.: TCP SYN attack on port 80 at victim

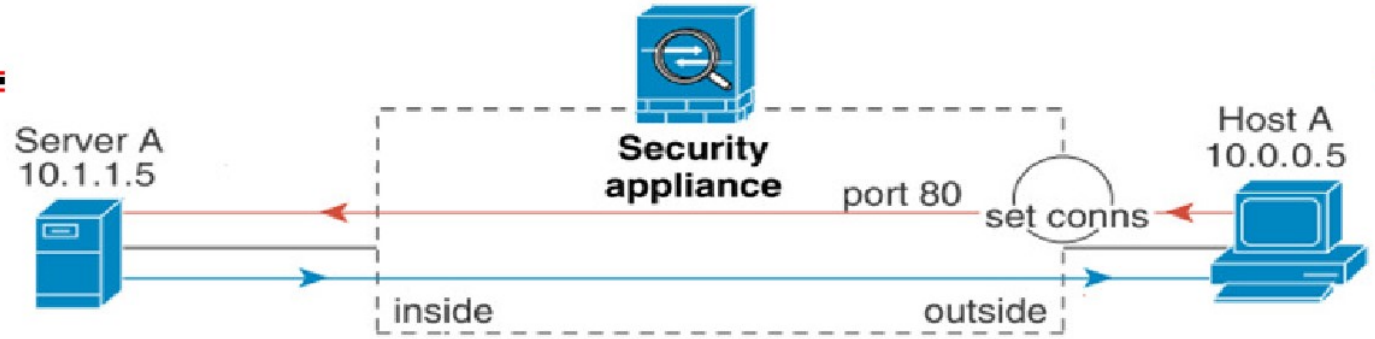
```
root@kali:~# hping3 -S -P --flood -V --rand-source -p 80
target_domain_name_or_ip
```

TCP SYN attack on XAMPP web server:



- Notice all TCP connections for the web server are stuck at SYN_RCVD state, consuming all system resources.
- Latest version of XAMPP may not crash, but you may notice system becomes sluggish during attack.

To defend TCP SYN attack, ASA can be configured using MPF to limit the maximum number of embryonic connections.



Step 1: Create class map to specify server to protect

```
ciscoasa(config)# access-list serverA extended permit tcp any host 10.1.1.5 eq 80
ciscoasa(config)# class-map http_serverA
ciscoasa(config-cmap)# match access-list serverA
```

Step 2: Create policy map to specify actions to take

10 and 100 are just examples

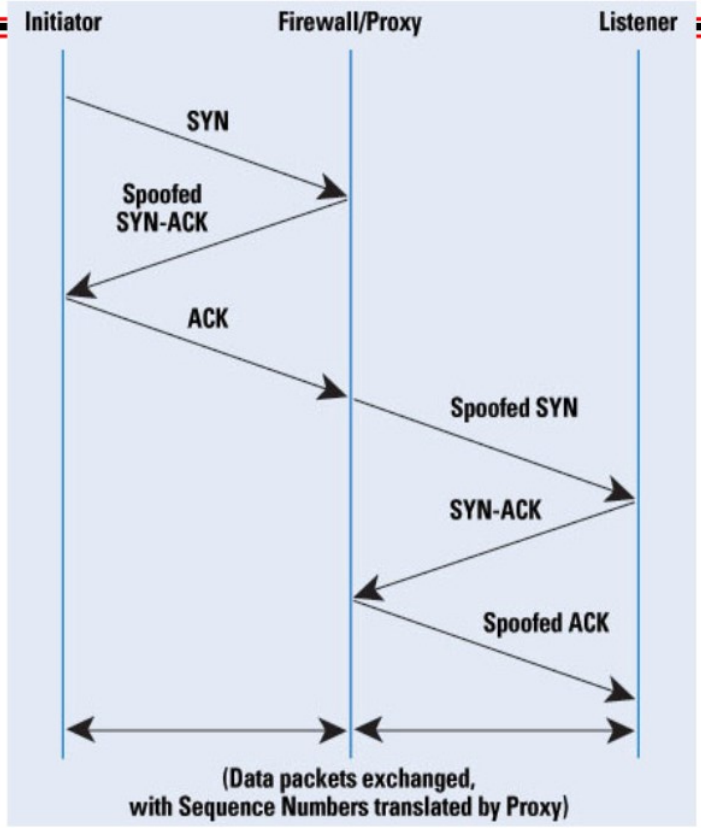
```
ciscoasa(config)# policy-map policy_serverA
ciscoasa(config-pmap)# class http_serverA
ciscoasa(config-pmap-c)# set connection embryonic-conn-max 100
ciscoasa(config-pmap-c)# set connection per-client-embryonic-max 10
```

Step 3: Apply policy map on appropriate interface

```
ciscoasa(config)# service-policy policy_serverA interface outside
```

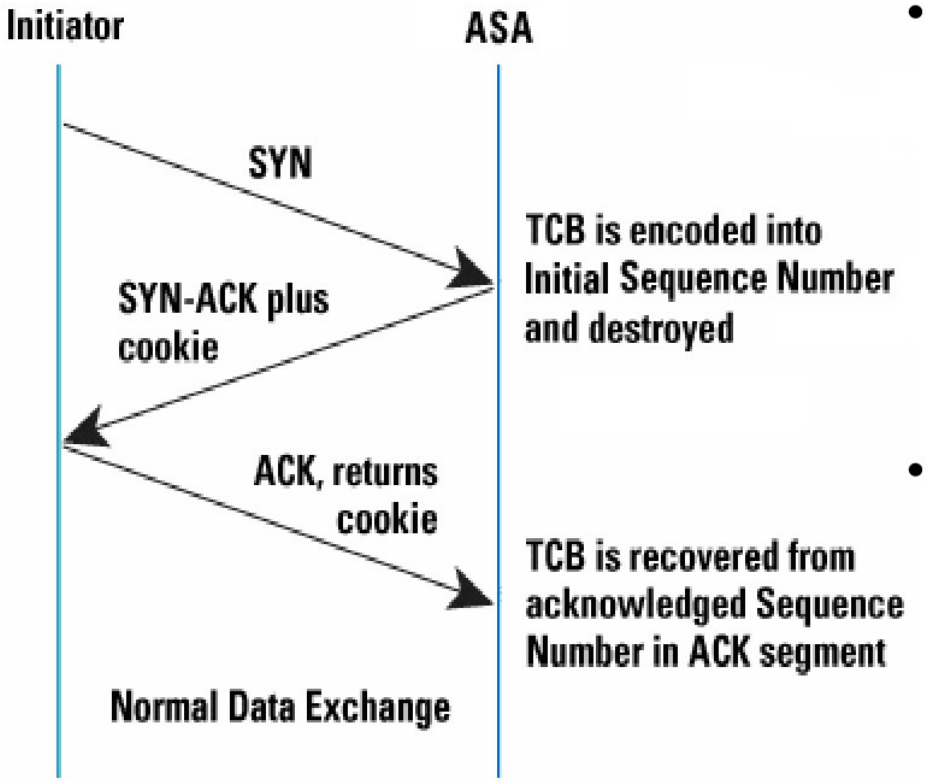
When the embryonic connection threshold is crossed, the ASA has a component called TCP Intercept which will be activated to act as a proxy for the server as follows:

- Intercept external SYN request to server and reply spoofed SYN-ACK on behalf of server.
- If subsequent ACK is received which indicates non-malicious request, then initiate spoofed SYN request to server to setup connection.



The maximum concurrent and embryonic connections can exceed the configured numbers, up to *n*-1 extra connections and embryonic connections, where *n* is the number of cores on your ASA.

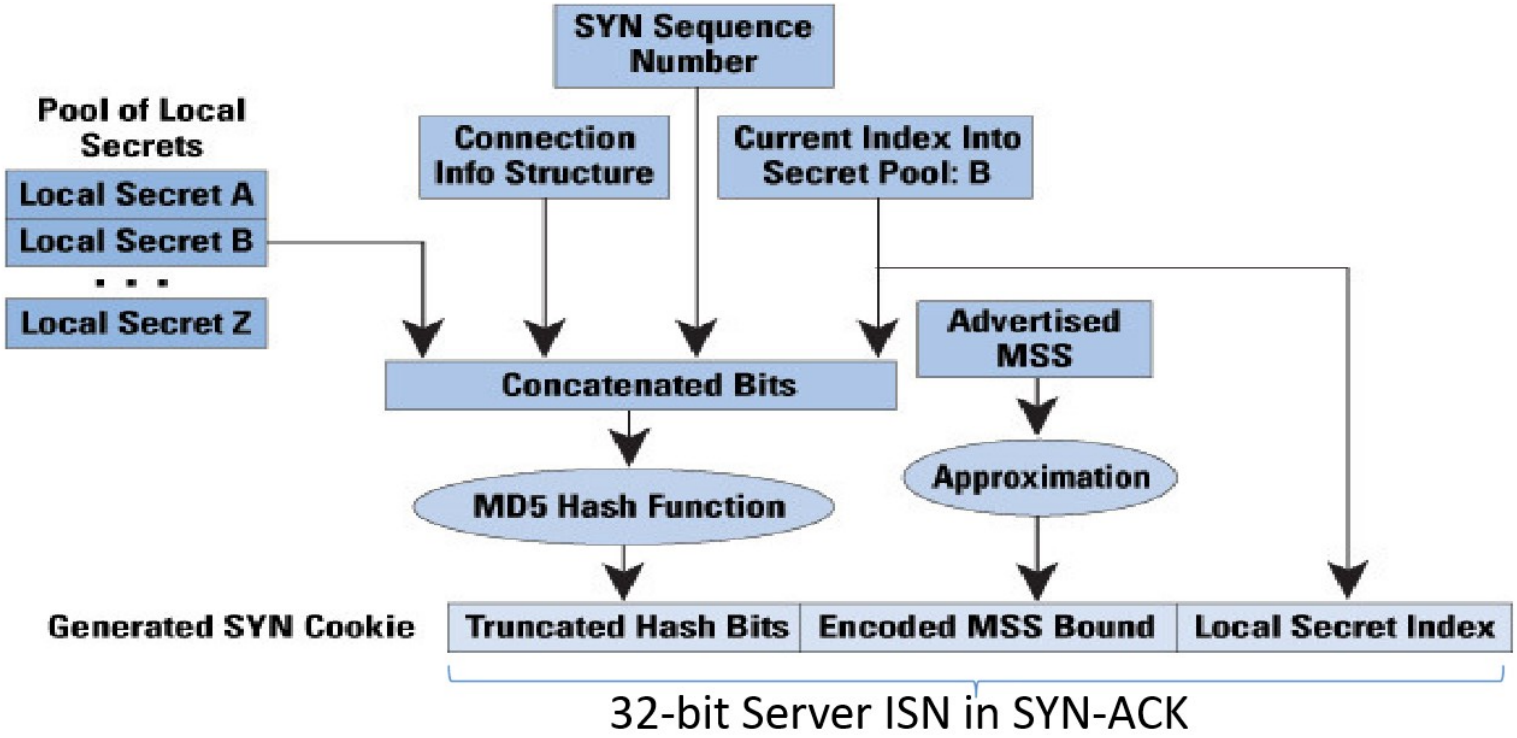
In addition, ASA implements TCP SYN cookie mechanism to protect itself as a proxy from TCP SYN flooding attack.



- The TCP TCB (Transmission Control Block), i.e. data structure/memory used to store TCP half-open state in 3-way handshake is encoded into ISN and destroyed.
- The reply, if arrives, will contain ACK sequence number ISN+1 which will enable proxy to reconstruct TCB, and if valid, to continue with the connection request.

The procedure for generating SYN cookie varies with different implementations, an example as follows:

- Connection Info Structure: IP addresses and TCP ports
- SYN Sequence Number: 32-bit Client ISN



- To validate reply, decrease ACK number – 1 to recover SYN cookie, and local secret index, then apply the same procedure. If valid, allow TCP connection to internal server.

To verify and monitor attacks intercepted by TCP intercept, you may configure threat detection statistics as follows:

- Enabling threat detection statistics for TCP intercept:

```
ciscoasa(config)# threat-detection statistics tcp-intercept
rate-interval 3 burst-rate 60 average-rate 30
```

- rate-interval sets the size of monitoring window in mins during which ASA samples the attacks 60 times
- burst-rate and average-rate are in attacks/events per sec (eps), when exceeded will generate syslog message
- Monitoring TCP intercept statistics:

```
ciscoasa(config)# show threat-detection statistics top tcp-intercept
Top 10 protected servers under attack (sorted by average rate)
Monitoring window size: 3 mins      Sampling interval: 3 secs
<Rank> <Server IP:Port> <Interface> <Ave Rate> <Cur Rate> <Total> <Source IP (Last Attack Time)>
------------------------------------------------------------------------------
1    10.1.1.5:80  inside  1249  9503  2249245  10.0.0.3 (0 secs ago)
```

Lab Exercise 3

SINGAPORE INSTITUTE OF TECHNOLOGY

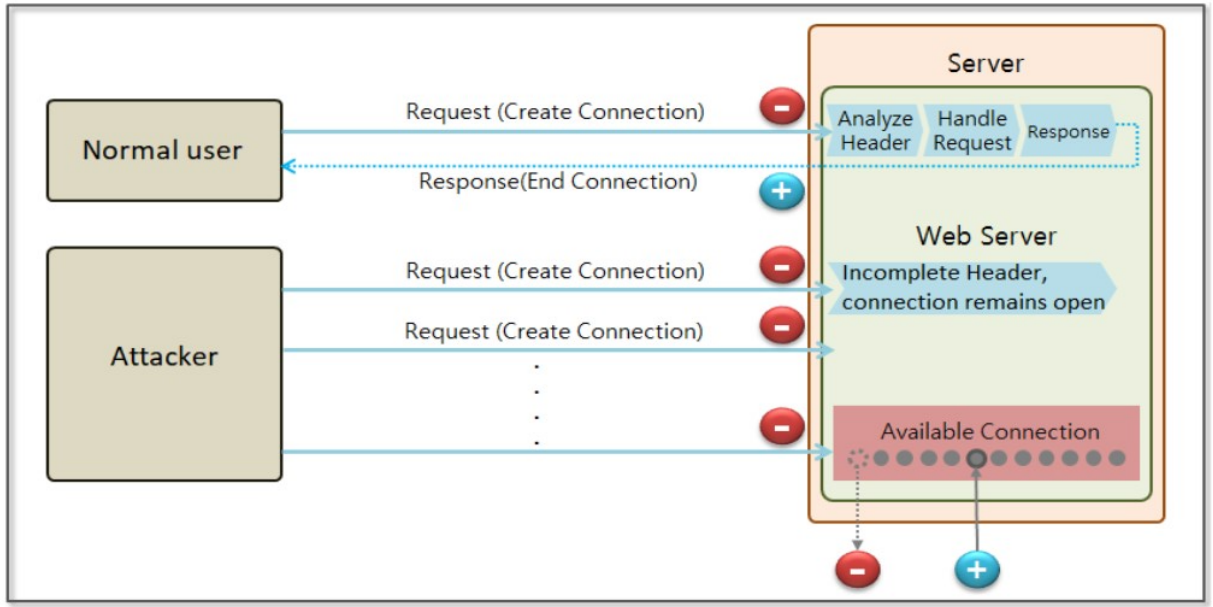
3.1 Slowloris attack

3.2 Defense:

- Configure MPF service policy in ASA

Slowloris attack is an example of DoS attack on web servers using low-rate but maliciously crafted packets to exploit design weakness of HTTP protocol.

By design, web server will only process HTTP request when header is received completely.



By creating many concurrent connections to the web server, and for each connection, intentionally sending HTTP header slowly in many small fragments, Slowloris is thus able to consume all connections and make the server unavailable.

To try Slowloris attack in the lab, you may use the slowhttptest in Kali Linux.

```
root@kali:~# slowhttptest -c number_connections -u target_url
```

Refer *https://tools.kali.org/stress-testing/slowhttptest* on the use of slowhttptest.

An example of Slowloris attack on XAMPP Apache web server with default setting.

Notice that a single client making 150 connections is sufficient to make Apache web server unavailable!

Similar as defending TCP SYN attack, MPF inspection security policy may be configured to defend Slowloris. Try it yourself!