Chapter 2

First-Seen-SYN Filter

2.1 SYN Flood

The rapid growth and increasing utilization of the Internet are making the networking security issue more and more important. One of the major problems is finite resources consumed by denial-of-service (DoS) which causes flooding packets [12, 25, 34]. DoS attack is easy to be issued but difficult to response since the number of flooding packets is usually large and telling the malicious packets from the legitimate ones is quite painful. The so-called Distributed Denial of Service attack (DDoS) stroke a huge number of popular web sites including Yahoo, Amazon, and others. The DDoS is called “distributed” because multiple distributed nodes attack a target server concurrently. Even if the attacking rate for each node is small, the attack traffic can still cause serious damage at the victim damage when the number of attacking nodes is large. There’re many kinds of DoS and DDoS attacks such as Smurf attacks, UDP attacks, TCP attacks, ICMP attacks, TCP SYN attacks [34]. Smurf and ICMP attack flood victim server’s link with a huge number of ICMP packets while UDP attacks generate many UDP packets to exhaust the victim server’s bandwidth. TCP floods are similar to UDP floods, except it uses TCP packets instead of UDP ones. TCP SYN attack takes advantage of the flaw of TCP three-way handshaking behavior by sending so many connection requests to a server so that it cannot accept more legitimate request anymore. Because it needs much less TCP SYN packets to exhaust a victim server’s resource than flooding its link by a large number of UDP packets, about 90% of all DoS attacks are TCP SYN attacks [53].

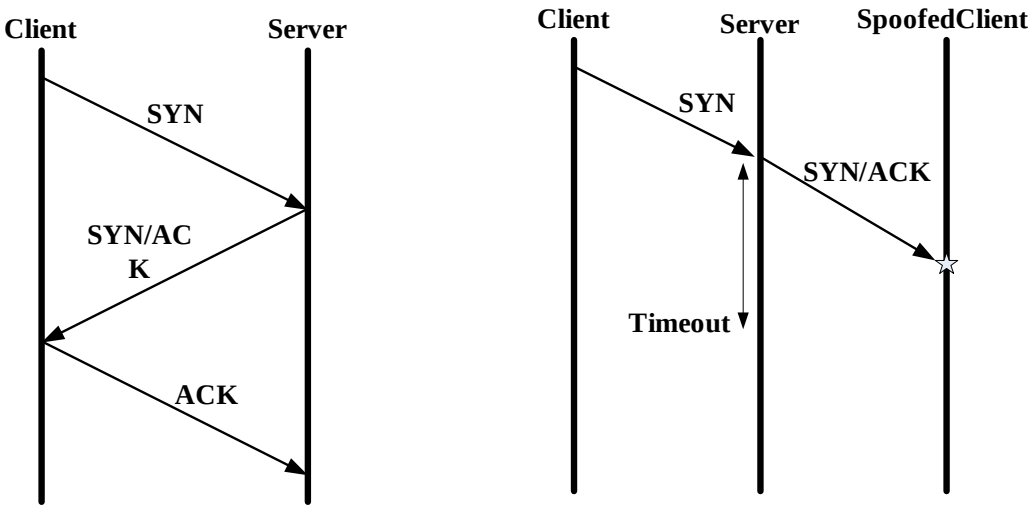


Figure 1. (a) TCP Three-Way Handshaking and (b) Spoofed SYN attack

A TCP three-way-handshaking process for establishing connections is shown in Figure 1 (a). A client first sends a SYN packet to server's listening port to request for connection establishment, and the server then allocates a space in the so-called backlog queue to store the connection state and related information. The server sends the client a SYN/ACK packet acknowledging its receipt of the SYN packet then, and waits for an ACK packet sent from the client for acknowledging receipt of the SYN/ACK packet and complete the three-way handshaking. The goal of a SYN flood is to exhaust the backlog queue in server side so that it cannot respond any further legitimate connections. This is achieved by making server keep waiting for the ACK packet which will never appear (as Figure 1 (b) shows) which results in the server retaining the backlog entry allocated for the initial SYN. Since the number of connections a server can maintain is bounded on the number of entries of the backlog queue, SYN packets in excess of the size of backlog queue are discarded and the server sends RST packets to clients for notification.

One of the problems in detecting SYN Flood traffic is that the packets used in SYN Flood attacks do not differ from normal TCP SYN packets except the source IP addresses may be spoofed [25]. This property makes server nodes or firewalls unable

to distinguish the SYN packets of normal traffic from those of SYN Flood attack. A simple response to SYN Flood traffic is the “first-seen packet reject” method: The first SYN packet sent by every untrustful host is discarded. Although one might challenge that the attacker might send each SYN packet twice, this works show that this simple design can overcome common SYN Flood tools efficiently and it can work in conjunction with other SYN Flood defenders to provide more protection.

The SYN flooding attack came out of the underground and into the public spotlight the first week of 1996 on an attack on the New York-based ISP Panix Public Access Network Corp. Because attacks can easily put servers into a DoS state this way, about 90% of all DoS attacks are SYN Flood attacks [53]. Therefore, many methods to defend servers from these attacks have been proposed.

2.2 Defending Mechanisms

In the ingress filtering [12], routers should drop packets with IP addresses out side the range of a customer’s network, so that they can prevent attackers from spoofing source IP addresses to launch a DoS attack. This method, however, should work in conjunction with other mechanisms because it does nothing to address flooding attacks that originate from valid IP addresses.

SYN Proxy [43] is widely adapted by both commercial and open-source firewalls to against SYN Flood attacks. In this approach, the firewall responds on the same behavior of the internal host. As Figure 2 (a) shows, the firewall responds the SYN packet for internal hosts instead of forwarding the SYN. If the SYN packet is sent by an attacker, there will be no more respond from the attacker, and the firewall will send RST packet to terminate the connection. On the other hand, if the initial SYN is sent by a legitimate user, the three-way handshaking will complete between the client and the firewall. Then the firewall creates a new connection to the internal host on the

same behavior of the client, and continues to act as a proxy for translating consequent sequence numbers of packets communicating between the client and a server. The strength of this method is that hosts never receive spoofed SYN packets. The main drawback of SYN Proxy method is its computing-intensive and serious memory consumption. Since it acts like an internal server, it has to maintain a huge number of TCP state entries to record each connection, and it needs to recalculate the checksum in translating the sequence number and building a SYN/ACK packet during the connection establishment. Another issue of this method is the difficulty to support TCP options. Since the firewall responds SYN/ACK packet for internal server, it has to deal with the TCP option request/reply including MSS option, window size option, and so on. Without the detail information regarding internal servers such as their operating systems, the firewall cannot do anything but reject all the TCP options.

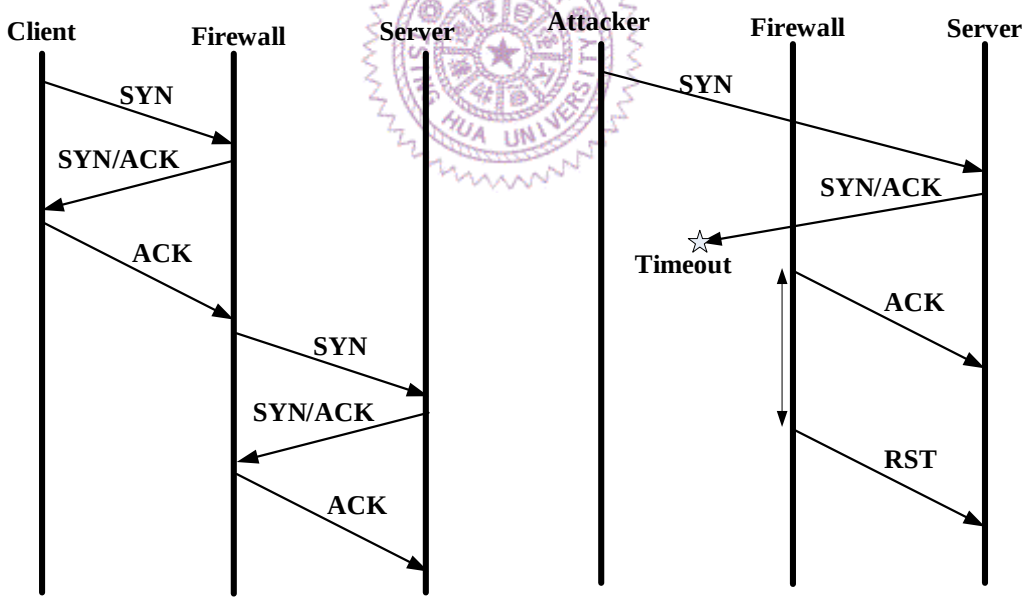
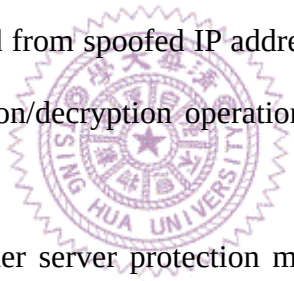


Figure 2. (a) SYN Proxy and (b) Semi-Transparent Gateway

Semi-Transparent Gateway [25] passes SYN packets to the host instead. As Figure 2 (b) shows, when the host responds SYN/ACK packet, the firewall forwards the packet to the client, and sends an ACK packet to the server to complete the three-way handshaking so that the backlog entry can be removed immediately. If the

firewall doesn't receive a legitimate ACK from the client after certain duration, a RST packet is sent to the host to terminate the connection. On the other hand, if the firewall does receive the legitimate ACK packet and pass to the server, the duplicate ACK is discarded with no harm. The strength of this method is no delays for legitimate connections, but if the number of incoming SYN packets exceeds the number of backlog entry within the timeout period, the SYN Flood attack still works.

SYN Cookies [4] is a server protection method in which the server encodes the source address, source port, source sequence, destination address, destination port, and a secret seed into the sequence number of the returning SYN/ACK packet. There will be no states stored in the server's side. If an ACK packet is received, the server verifies whether it's a legitimate response or not. The SYN Cookies does well in blocking TCP floods originated from spoofed IP addresses. However its problems are computing power for encryption/decryption operations and lack of supporting TCP options either.



SYN Cache [27] is another server protection method which allocates minimal state when the initial SYN is received, and it only allocates all the resources required when the three-way handshaking is completed. This method replaces the backlog queue with a global hash table, which provides two forms of protection against running out of resources. These are a limit on the total number of entries in the table, which provides an upper bound on the amount of memory that the SYN Cache takes up, and a limit on the number of entries in a given hash bucket. Every time the server receives a SYN packet, it allocates a hash entry for this connection instead of allocating a backlog entry. If the total number of entries in the hash table is exceeded, the oldest entry is dropped. Since a hash entry cost much less memory than TCP state entry and the hash table size can be very large, SYN Cache can reduce the impact of the TCP SYN Flood by this resource allocation strategy.

2.3 Design and Implementation of FSS Filter

This works relies on an observation over common SYN-flooding programs. SYN-flooding tools usually have good ability in spoofing the source IP addresses, randomizing the source port numbers, and even the TCP sequence numbers. Normally the source IP addresses are distributed quite uniformly. According to our observation, HGOD can generate millions of SYN packets without two duplicate source IP addresses. Furthermore, even when we constrain the source IP range in a class C domain, the source port numbers and the sequence numbers are dramatically randomized as well. The advantage of disguising attacking packets like this is to hide the real attacking source and keep routers, firewalls, and even ends systems from being aware of flooding packets generated by single hosts. By this way, there're barely two identical SYN packets appearing during attacking with this kind of tools. Real TCP connections retransmit though. If the first SYN packet is discarded, real TCP connections tend to retransmit another one after the round-trip-time (RTT) while most attacking tools have no idea about this. They have no way to know whether the first SYN packets have been dropped or not since they spoofed the source IP addresses so that they will not receive the responding SYN-ACK packets. We can make use of this characteristic to tell real traffic from the attacking traffic.

This design comprises four other tables except to ordinary TCP state table: FSS (First-Seen SYN) table records the appeared SYN packets, SSC (Per Source Session Count) table records the number of flows issued by hosts, THC (Trusted-Host Cache) table records the hosts which have been considered as good and unaffected ones, and DSC (Destination Session Count) table records how many SYN packets sent to per server in a certain duration. When we sense that there have been too many SYN packets sent to one server by looking up the DSC table, we will start to discard the

first-seen SYN packet against the victim server. Only the SYN packets (retransmission ones actually) appeared in FSS table can pass the filter. SSC table is used to constrain the number of flows issued by a single host. Sometimes the attacker doesn't spoof the source IP address of the attacking SYN packets and we can easily block those packets if we know exactly how many SYN packets are generated by this host. On the other hand, if the SYN packets somehow evade the FSS filter successfully, SSC is still a baseline defender. THC table can help us to identify the "good" hosts from the spoofed bad ones. Once a host is able to complete the three-way-handshaking operation and exchange data with the server, we will consider it as a good host and add its IP address to THC table so that we will not drop its SYN packets unless it transmits more than the threshold according to SSC table.

Figure 3 shows the processing procedure of the FSS filter. First it checks and updates the SSC table to make sure the number of SYN packets transmitted by the source doesn't exceed the pre-defined threshold. And then it checks and updates the DSC table to find out whether the destination host receives too many SYN packets in certain duration or not. If not, it builds a new entry in TCP State Table for the incoming flow. Otherwise, it checks whether the source IP address is a legitimate address or not so that we can decide to drop this packet and insert a corresponding entry in FSS table or simply build a new entry in TCP State Table for it. Other than the processing of SYN packet, this design needs another two more operations. When a client completes the three-way handshaking with a guarded server, we will build a trusted entry for this client in THC Table so that we will not drop its SYN packet again unless the amount exceeds the threshold. The FSS filter is mainly a pre-filter which should work in conjunction with other mechanisms. Later we will show that by combining the FSS filter with other well-known defenders has much more advantages in many aspects.

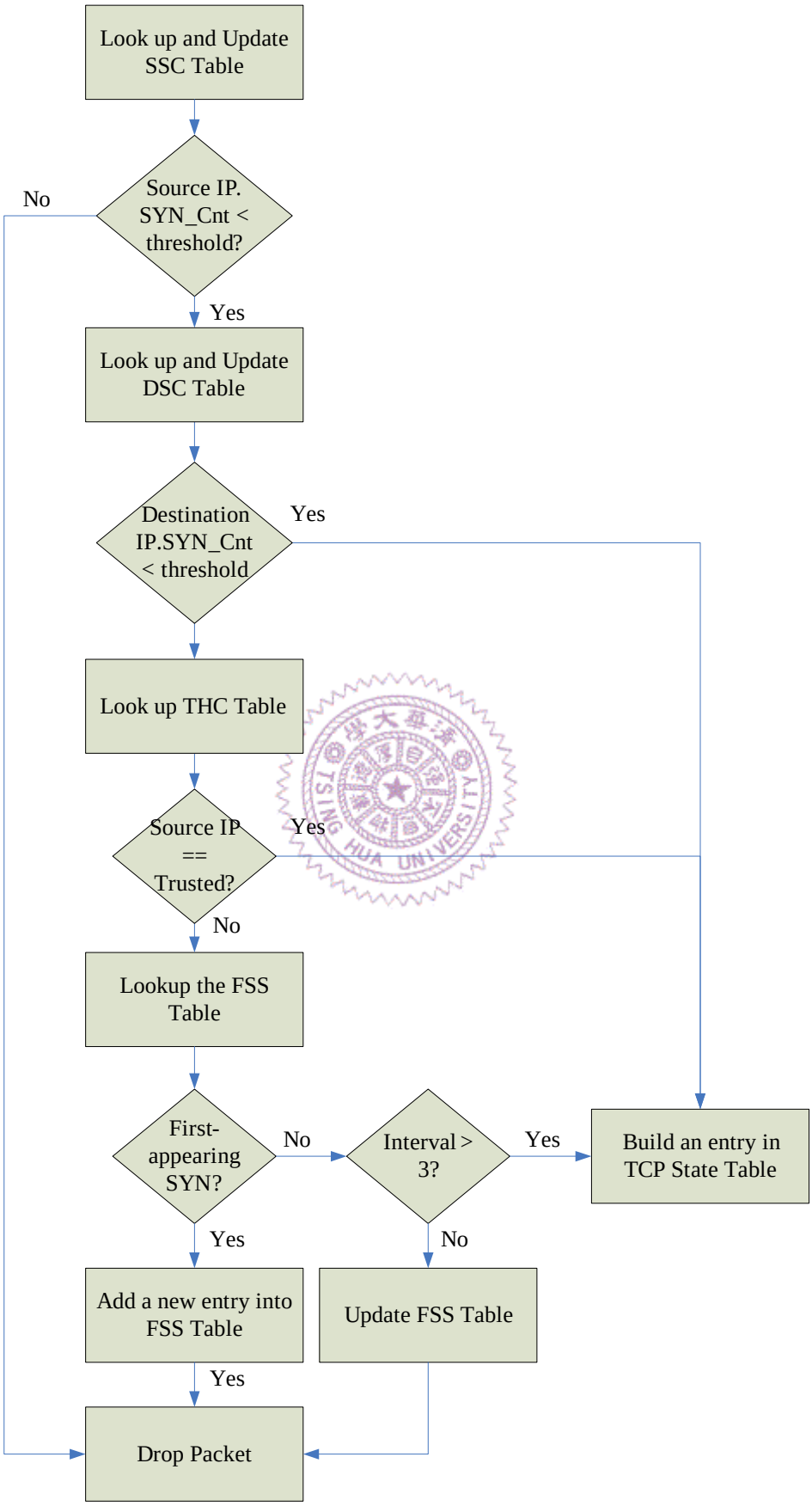


Figure 3. The processing flow of the FSS filter.

2.4 Analysis of FSS Filter

The main advantage of FSS filter is its tremendous ability in blocking IP-spoofed packets. However, the cost is relatively low comparing to other solutions. Every entry in FSS Table is 16 bytes which comprises 4-byte source IP address, 4-byte destination IP address, 2-byte source port, 2-byte destination port, 2-byte timestamp, and 2-byte of the last 16-bit of sequence number. Every entry in SSC Table is 8 bytes which comprises 4-byte source IP address, 2-byte timestamp, and 2-byte counter for counting number of sent SYN packets. Every entry in THC Table is 8 bytes which comprises 4-byte source IP address, 2-byte timestamp, and 2-byte counter for counting number of sent SYN packets by this IP. These are very light-weighted cost since a TCP state entry can be 280 bytes.

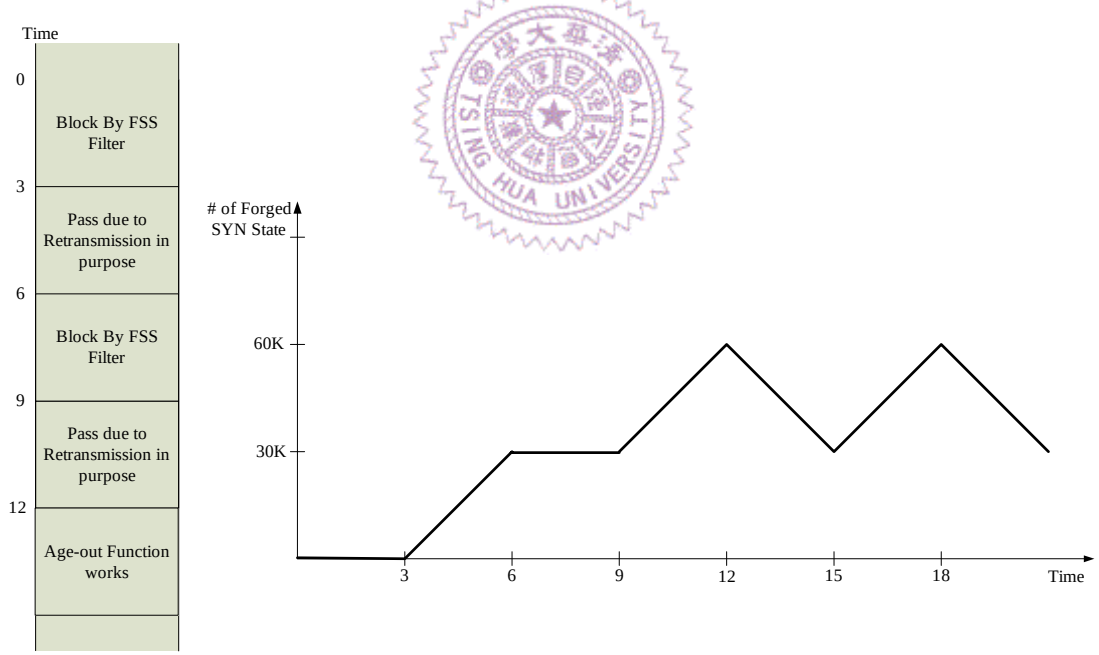


Figure 4. FSS filter co-works with Semi-Transparent Gateway

Recent experiments with SYN flood on commercial platforms show that an attack rate of only 500 SYN packets per second is enough to overwhelm a server which an attack rate of 14,000 packets per second can disable the ability of SYN proxy of a specialized firewall. There’re two obvious reasons for the number 14,000.

One is insufficient computing power to establish a state (a lot of memory read/write for initialization) and send back a SYN-ACK packet (involving buffer copying, checksum calculating, and so on). For example, the connection setup rate of a leading commercial firewall which accommodates more than 400Mbps traffic is only 11,500 per second. This issue is getting worse under SYN flood because the 11,500 connection includes both the forged SYN packets and legitimate connections. The server under attack cannot serve normal users in the same rate as it does in normal situation. The second issue is insufficient memory spaces. Considering that the SYN-Proxy-based firewall will keep receiving the SYN packets and establish corresponding connection entries, and it will monitor the state of these entries and flush them out if the further ACK packet is not received to complete the three-way handshaking. Assume that its age-out time is 10 seconds, then it needs another 140,000 entries for processing the forged SYN packets and it may affect the normal operation since a hundred-Mbps-level firewall usually supports only 65,536 concurrent sessions. On the other hand, we can have 128K-entry FSS Table, 32 K SSC, and 4K THC which costs 2,336 KB only and can sufficiently block the SYN flood if the attacking programs don't retransmit in purpose. Besides, the FSS filter is not computation-intensive at all comparing to expensive TCP state maintenance and SYN-ACK packets transmission. The tables can be implemented by a simple four-way hashing tables since the forged packets are as randomized as possible, it's most likely that the FSS entries are uniform distributed also. With the help of FSS filter, the SYN-Proxy server can still sustain its 11.5K connection setup rate for legitimate users. Even the attackers are aware of the FSS filter and retransmit the forged SYN packets in purpose, the overall attack rate drops down to 50% only. The following example of integrating FSS filter with semi-transparent-gateway mechanism explains how the FSS filter blocks at least 50% malicious SYN packets.

Combining FSS filter with Semi-Transparent Gateway also provides better level of defenses than Semi-Transparent Gateway only. Figure 4 shows how the TCP state entries of the protected server guarded by FSS filter are occupied for forged SYN packets during attack. Assume that FSS filter will only accept retransmission packets after 3-second interval and the age-out time of non-acknowledged entry is 9 second for simplicity. The incoming rate of forged SYN packets is set to 10K per second. Before 3rd second after the flood issues, all the forged SYN packets are blocked. From 3rd second to 6th second, if the attacking tool retransmits the forged packets in purpose, we will pass it into server side. From 6th second to 9th second, we will still block all the forged SYN packets. And from 9th second to 12th second, we will pass the attacked SYN packets again. But we will start to clean up the zombie entries generated from 3rd second to 6th second since they're aged-out. If the attack remains, the number of stalled state entries of the protected server may range from 30K to 60K back and forth. Figure 5 shows how Semi-Transparent Gateway alone works under attack. It will start to clean the stalled TCP state entries after 9 seconds, but since the attack is continuing in the same time, the number of stalled state entries of the protected server may remain. Under the mentioned assumptions FSS filter saves 50% TCP state entries compared to original design.

Moreover, integrating FSS filter with SYN-Cache mechanism performs better than utilize SYN-Cache alone. The main idea of SYN-Cache is to adapt a hash table to store the half-opened connections to prevent the system from allocating too many memories for the forged SYN packets because the size of its hash entry is around 20% of original TCP state entry. However, the flood still succeeds when the number of forged SYN packets exceeds the total hash entries. The FSS filter can help SYN-Cache to defense more flooding packets to 200%. Originally a 128K-entry SYN-Cache can handle at most 128K forged SYN packets, but it can accommodate at

least 256K forged SYN packets with the help of FSS filter because the attackers need to retransmit the forged packets and therefore the attacking rate drops down to 50%. For example, if the forged SYN packets flood in the rate of 32K packets per second (consuming 22Mbps) the system can sustain 8 seconds until the age-out mechanism starting to clean up the zombie entries. Providing this level of defense works in most cases since common DSL lines available are less than 22Mbps, if the attacking rate exceeds this number than it's viewed as a bandwidth consuming problem which cannot be solved essentially.

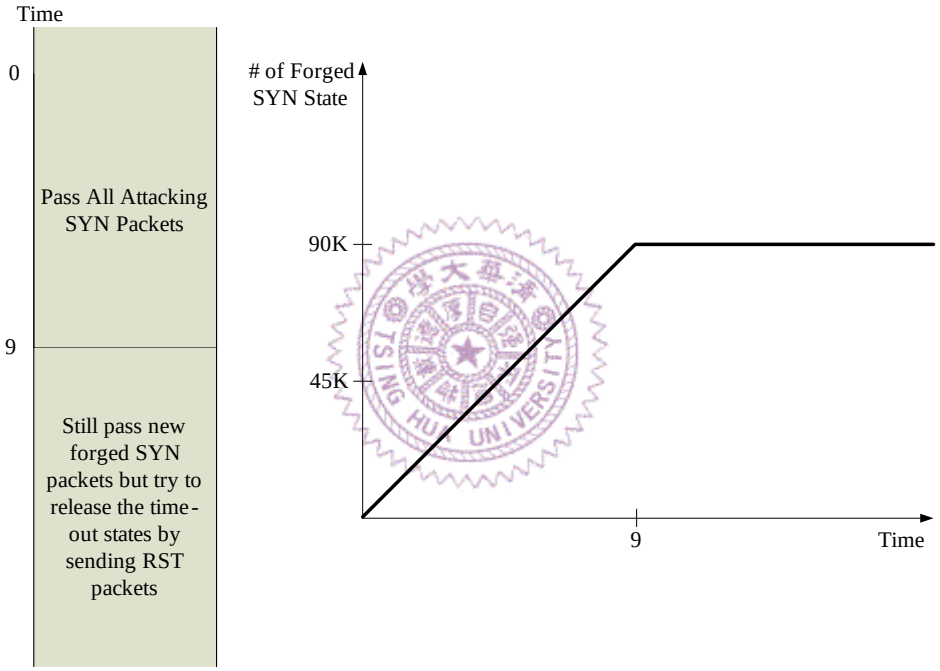


Figure 5. Semi-Transparent Gateway works alone.

2.5 Experiments over FSS Filter

Figure 6 shows the test environment of FSS filter. A Web server is guarded by FSS filter. The Web Server is a general PC with Pentium IV 2.0GHz processor and 512MB DDR SDRAM. The web service runs on Linux Apache web server. The hardware platform of the FSS filter is an Industrial PC (IPC) with a Pentium III 800MHz processor and 256MB DDR SDRAM on it. The operating system is pSOS+, a stable-proven Realtime OS (RTOS) which we can easily insert our implementation

into. In this test, the size of FSS table is 128K, and the size of SSC table is 32K, and THC table's size is 4K. The memory consumption is 2MB for FSS, 256KB for SSC, and 32KB for THC, which costs 2,336 KB in total.

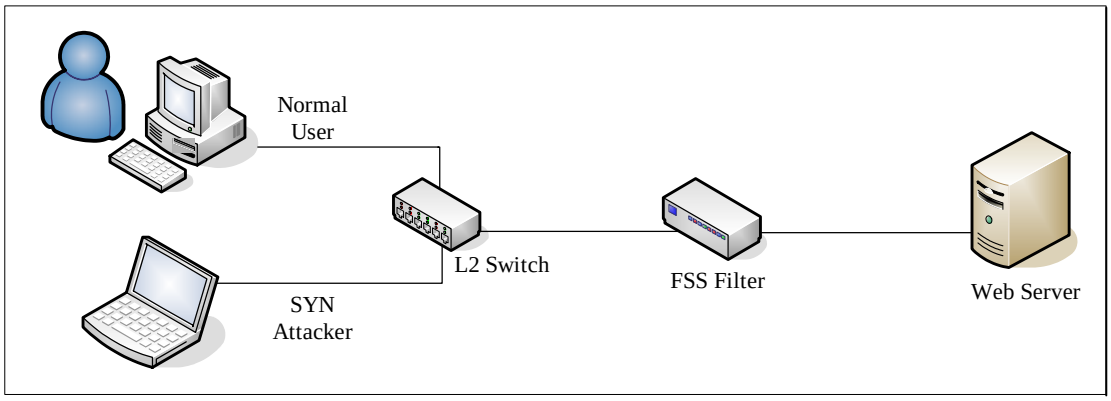


Figure 6. Test environment of FSS Filter

The FSS filter is running on inline mode (bridge mode) and, in addition to the web server, another end is a Layer-2 switch with two PCs connected to it. One PC is to establish normal requests to the web server, and the other one is to generate the SYN Flood traffic against to the same web server. The attacker PC is running on Linux operating system so that we can find a bunch of attacking tools to test, it can also boot under Windows XP professional edition since some attacking tools (e.g. HGOD) are win32 applications.

Table I. Attacking tool's list in the FSS Filter experiment

Attack Tools	Description
3wahas	LAN based syn flooder which spoofs syn ACK packets too, allowing to bypass syn-cookies.
Punk	syn Flooder source code with spoofed source address, much like slice2.
Blitznet	Blitznet launches a distributed syn flood attack with spoofed source IP, without logging.
Xcrush-20	Latest version of the infamous war program. Includes Beer, Biffit, Boink, Bonk, Dcd3c, Foqer, Gewse5, Ghost, Hanson, Hestra, Ice, ICQCrash, ICQFlood, ICQSpoof, IRCd Kill, Jolt, Land, mIRCKill, Mutilate, Nestea, Newtear, Octopus, OOB,

	Overdrop, Pepsi, Pong, Rape, ServUKill, SSping, syndrop, synflood, Teardrop, and WarFTPd Kill. Requires Tcl/Tk.
Poseidon	Poseidon++ syn flooder, complete with tcl/tk gui, by daemon9.
Synful	syn (SYN/ACK and ACK blow) written by \\StOrM\\
Gateway	Password protected remote shell daemon that integrates a syn flooder, bouncer/gateway, port scanner, and remote root exploits. Courtesy of Mixter.
Tfn	Distributed flood network client/server that can be installed on a large number of hosts and used to hit a target with high bandwidth simultaneously. communicates over icmp and supports udp, syn, icmp/8, smurf flood and more. Courtesy of Mixter.
Apsend	APSEND v1.60 is a TCP/IP packet sender to test firewalls and other network applications. Includes a syn flood option, land DoS attack, dos attack against tcpdump 3.4, and spoofing. You can also choose many other options, ie TTL, ToS, sequence number, ack number, urgent pointer, syn/PUSH/ACK/RST/URG/FIN flag, window size, and number of packets to send.
Dos.pl	DoS.pl uses Net::RawIP to launch a syn flood attack.
Datapool3.3	Datapool v3.3 combines 106 dos attacks into one script. This version actually learns by keeping a database of which attacks are successful against each host, so the next time it uses the most successful attack first. Features logging, port rance specification, continous attack option, multiple IP addresses, and looping attack of multiple IPs. Includes sources of almost all attacks used, many of which are edited for speed and greater effect.
Juno	juno.c is a tcp syn flooder with extremely fast packet creation routines and the ability to emulate Linux or Windows.
Knight	knight is a distributed denial of service client that is very light weight and is very powerful. It goes on IRC and joins a channel, then accepts commands via IRC (to prevent from getting caught). It has features like, an automatic updater via http or ftp, a checksum generater, a syn flooder, a tcp flooder, a udp flooder, slice2, spoofing to subnets, and more. This program has been used to create DDoS nets of over 1000 clients.
Netflood	Netflood.c is useful for testing spoof rules on gateways, testing syn flood defense mechanisms/configurations (like Checkpoint's

	syndefender module), testing IDS syn flood/Land attack signatures. It can, of course, be used for engaging in syn Flood attacks and Land attacks. It also counts number of packets sent.
Flood2.c	Flood2.c is a syn flooder that is more efficient than Juno because it uses smaller packets. Slightly broken.
Imp	Imp is a denial of service tool which sends syn floods. Some people call this one slice3. Dynamically linked with libc5.
Myndscream	Myndscream is a syn flooder.
HGod	Hgod is a small denial of service tool which runs on Windows 2000 and XP and sends several types of packet floods

Table I shows the attacking tools adapted in this test. The test procedure is as follows. First we enable the FSS Filter and execute the SYN Flooding tool on the attacker PC. The threshold of the FSS filter is set to 300, i.e. it starts to block first-seen SYN packets when receiving over 300 SYN packets per second. Then we established web connections to the victim web server from normal user's browser. If the FSS filter is disabled or absent, we cannot connect to the victim server successfully because it cannot accept any legitimate connections anymore. In our test, FSS filter can block all the SYN Flooding attacks generated by the collected tools efficiently and protect the victim web server from exhausting its precious state resources.

In summary, FSS filter is a low-cost and efficient mechanism to protect servers from being attacked by SYN Flooding tools. In our test, all the attacking tools failed to exhaust the victim server's TCP state so that the availability of its web service is assured. We cannot find even one attacking tool against the weakness of FSS filter, but if there does exist, this mechanism can still reduce at least 50% attacking rate so that it can work in conjunction with other defense mechanisms such like SYN-Cache, SYN-Proxy, and so on. The extra memory space for FSS filter is much relatively small than the others, and this design requires much less computing power as well. The drawback of this mechanism is dropping the first SYN packet of even legitimate

clients, but it only happens once by the help of THC table.