

## Hardening the TCP/IP stack to SYN attacks

*Mariusz Burdach* 2003-09-10

Most people know how problematic protection against SYN denial of service attacks can be. Several methods, more or less effective, are usually used. In almost every case proper filtering of packets is a viable solution. In addition to creating packet filters, the modification of the TCP/IP stack of a given operating system can be performed by an administrator. This method, the tuning of the TCP/IP stack in various operating systems, will be described in depth in this article.

While SYN attacks may not be entirely preventable, tuning the TCP/IP stack will help reduce the impact of SYN attacks while still allowing legitimate client traffic through. It should be noted that some SYN attacks do not always attempt to upset servers, but instead try to consume all of the bandwidth of your Internet connection. This kind of flood is outside the scope of scope of this article, as is the filtering of packets which has been discussed elsewhere.

What can an administrator do when his servers are under a classic, non-bandwidth flooding SYN attack? One of most important steps is to enable the operating system's built-in protection mechanisms like SYN cookies or `SynAttackProtect`. Additionally, in some cases it is worth tuning parameters of the TCP/IP stack. Changing the default values of stack variables can be another layer of protection and help better secure your hosts. In this paper I will concentrate on:

- Increasing the queue of half-open connections (in the SYN RECEIVED state).
- Decreasing the time period of keeping a pending connection in the SYN RECEIVED state in the queue. This method is accomplished by decreasing the time of the first packet retransmission and by either decreasing the number of packet retransmissions or by turning off packet retransmissions entirely. The process of packet retransmissions is performed by a server when it doesn't receive an ACK packet from a client. A Packet with the ACK flag finalizes the process of the three-way handshake.

Note that an attacker can simply send more packets with the SYN flag set and then the above tasks will not solve the problem. However, we can still increase the likelihood of creating a full connection with legitimate clients by performing the above operations.

We should remember that our modification of variables will change the behavior of the TCP/IP stack. In some cases the values can be too strict. So, after the modification we have to make sure that our server can properly communicate with other hosts. For example, the disabling of packet retransmissions in some environments with low bandwidth can cause a legitimate request to fail.

In this article you will find a description of the TCP/IP variables for the following operating systems: Microsoft Windows 2000, RedHat Linux 7.3, Sun Solaris 8 and HP-UX 11.00. These variables are similar or the same in current releases.

## Definitions: SYN flooding and SYN spoofing

A SYN flood is a type of Denial of Service attack. We can say that a victim host is under a SYN flooding attack when an attacker tries to create a huge amount of connections in the SYN RECEIVED state until the backlog queue has overflowed. The SYN RECEIVED state is created when the victim host receives a connection request (a packet with SYN flag set) and allocates for it some memory resources. A SYN flood attack creates so many half-open connections that the system becomes overwhelmed and cannot handle incoming requests any more.

To increase an effectiveness of a SYN flood attack, an attacker spoofs source IP addresses of SYN packets. In this case the victim host cannot finish the initialization process in a short time because the source IP address can be unreachable. This malicious operation is called a SYN spoofing attack.

We need to know that the process of creating a full connection takes some time. Initially, after receiving a connection request (a packet with SYN flag set), a victim host puts this half-open connection to the backlog queue and sends out the first response (a packet with SYN and ACK flags set). When the victim does not receive a response from a remote host, it tries to retransmit this SYN+ACK packet until it times out, and then finally removes this half-open connection from the backlog queue. In some operating systems this process for a single SYN request can take about 3 minutes! In this document you will learn how to change this behavior. The other important information you need to know is that the operating system can handle only a defined amount of half-open connections in the backlog queue. This amount is controlled by the size of the backlog queue. For instance, the default backlog size is 256 for RedHat 7.3 and 100 for Windows 2000 Professional. When this size is reached, the system will no longer accept incoming connection requests.

## How to detect a SYN attack

It is very simple to detect SYN attacks. The netstat command shows us how many connections are currently in the half-open state. The half-open state is described as SYN\_RECEIVED in Windows and as SYN\_RECV in Unix systems.

```
# netstat -n -p TCP
```

```

tcp      0      0 10.100.0.200:21          237.177.154.8:25882      SYN_RECV  -
tcp      0      0 10.100.0.200:21          236.15.133.204:2577     SYN_RECV  -
tcp      0      0 10.100.0.200:21          127.160.6.129:51748     SYN_RECV  -
tcp      0      0 10.100.0.200:21          230.220.13.25:47393     SYN_RECV  -
tcp      0      0 10.100.0.200:21          227.200.204.182:60427   SYN_RECV  -
tcp      0      0 10.100.0.200:21          232.115.18.38:278       SYN_RECV  -
tcp      0      0 10.100.0.200:21          229.116.95.96:5122      SYN_RECV  -
tcp      0      0 10.100.0.200:21          236.219.139.207:49162   SYN_RECV  -
tcp      0      0 10.100.0.200:21          238.100.72.228:37899    SYN_RECV  -
...

```

We can also count how many half-open connections are in the backlog queue at the moment. In the example below, 769 connections (for TELNET) in the SYN RECEIVED state are kept in the backlog queue.

```

# netstat -n -p TCP | grep SYN_RECV | grep :23 | wc -l
769

```

The other method for detecting SYN attacks is to print TCP statistics and look at the TCP parameters which count dropped connection requests. While under attack, the values of these parameters grow rapidly.

In this example we watch the value of the `TcpHalfOpenDrop` parameter on a Sun Solaris machine.

```

# netstat -s -P tcp | grep tcpHalfOpenDrop
tcpHalfOpenDrop      =      473

```

It is important to note that every TCP port has its own backlog queue, but only one variable of the TCP/IP stack controls the size of backlog queues for all ports.

## The backlog queue

The backlog queue is a large memory structure used to handle incoming packets with the SYN flag set until the moment the three-way handshake process is completed. An operating system allocates part of the system memory for every incoming connection. We know that every TCP port can handle a defined number of incoming requests. The backlog queue controls how many half-

open connections can be handled by the operating system at the same time. When a maximum number of incoming connections is reached, subsequent requests are silently dropped by the operating system.

As mentioned before, when we detect a lot of connections in the SYN RECEIVED state, host is probably under a SYN flooding attack. Moreover, the source IP addresses of these incoming packets can be spoofed. To limit the effects of SYN attacks we should enable some built-in protection mechanisms. Additionally, we can sometimes use techniques such as increasing the backlog queue size and minimizing the total time where a pending connection is kept in allocated memory (in the backlog queue).

## Built-in protection mechanisms

### Operating system: Windows 2000

The most important parameter in Windows 2000 and also in Windows Server 2003 is `SynAttackProtect`. Enabling this parameter allows the operating system to handle incoming connections more efficiently. The protection can be set by adding a `SynAttackProtect` DWORD value to the following registry key:

```
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
```

In general, when a SYN attack is detected the `SynAttackProtect` parameter changes the behavior of the TCP/IP stack. This allows the operating system to handle more SYN requests. It works by disabling some socket options, adding additional delays to connection indications and changing the timeout for connection requests.

When the value of `SynAttackProtect` is set to 1, the number of retransmissions is reduced and according to the vendor, the creation of a route cache entry is delayed until a connection is made. The recommended value of `SynAttackProtect` is 2, which additionally delays the indication of a connection to the Windows Socket until the three-way handshake is completed. During an attack, better performance in handling connections is achieved by disabling the use of a few parameters (these parameters are usually used by the system during the process of creating new connections). The `TCPInitialRTT` parameter, which defines the time of the first retransmission, will no longer work. It's impossible to negotiate the window size value. Also, the scalable windows option is disabled on any socket.

As we can see, by enabling the `SynAttackProtect` parameter we don't change the TCP/IP stack behavior until under a SYN attack. But even then, when `SynAttackProtect` starts to operate, the operating system can handle legitimate incoming connections.

The operating system enables protection against SYN attacks automatically when it detects that values of the following three parameters are exceeded. These parameters are `TcpMaxHalfOpen`, `TcpMaxHalfOpenRetried` and `TcpMaxPortsExhausted`.

To change the values of these parameters, first we have to add them to the same registry key as we made for `SynAttackProtect`.

The `TcpMaxHalfOpen` registry entry defines the maximum number of SYN RECEIVED states which can be handled concurrently before SYN protection starts working. The recommended value of this parameter is 100 for Windows 2000 Server and 500 for Windows 2000 Advanced Server.

`TcpMaxHalfOpenRetried` defines the maximum number of half-open connections, for which the operating system has performed at least one retransmission, before SYN protection begins to operate. The recommended value is 80 for Windows 2000 Server, and 400 for Advanced Server.

The `TcpMaxPortsExhausted` registry entry defines the number of dropped SYN requests, after which the protection against SYN attacks starts to operate. Recommended value is 5.

## **Operating system: Linux RedHat**

RedHat, like other Linux operating systems, has implemented a SYN cookies mechanism which can be enabled in the following way:

```
# echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Note that to make this change permanent we need to create a startup file that sets this variable. We must do the same operation for other UNIX variables described in this paper because the values for these variables will return to default upon system reboot.

SYN cookies protection is especially useful when the system is under a SYN flood attack and source IP addresses of SYN packets are also forged (a SYN spoofing attack). This mechanism allows construction of a packet with the SYN and ACK flags set and which has a specially crafted initial sequence number (ISN), called a cookie. The value of the cookie is not a pseudo-random number generated by the system but instead is the result of a hash function. This hash result is

generated from information like: source IP, source port, destination IP, destination port plus some secret values. During a SYN attack the system generates a response by sending back a packet with a cookie, instead of rejecting the connection when the SYN queue is full. When a server receives a packet with the ACK flag set (the last stage of the three-way handshake process) then it verifies the cookie. When its value is correct, it creates the connection, even though there is no corresponding entry in the SYN queue. Then we know that it is a legitimate connection and that the source IP address was not spoofed. It is important to note that the SYN cookie mechanism works by not using the backlog queue at all, so we don't need to change the backlog queue size. More information about SYN cookies can be found at <http://cr.yip.to/syncookies.html>.

Also note that the SYN cookies mechanism works only when the CONFIG\_SYNCOOKIES option is set during kernel compilation.

The next section will describe other useful methods of protection against SYN attacks. I would like to emphasize that under heavy SYN attacks (like Distributed SYN flooding attack) these methods may help but still not solve the problem.

## Increasing the backlog queue

Under a SYN attack, we can modify the backlog queue to support more connections in the half-open state without denying access to legitimate clients. In some operating systems, the value of the backlog queue is very low and vendors often recommend increasing the SYN queue when a system is under attack.

Increasing the backlog queue size requires that a system reserve additional memory resources for incoming requests. If a system has not enough memory for this operation, it will have an impact on system performance. We should also make sure that network applications like Apache or IIS can accept more connections.

### Operating system: Windows 2000

Aside from described above `TcpMaxHalfOpen` and `TcpMaxHalfOpenRetried` variables, in Windows 2000 the number of connections handled in the half-open state can be set through a dynamic backlog. Configuration of this dynamic backlog is accomplished via the AFD.SYS driver. This kernel-mode driver is used to support Windows Socket applications like FTP and Telnet. To increase the number of half-open connections, AFD.SYS provides four registry entries. All of these values, corresponding to AFD.SYS, are located under the following registry key:

HKLM\System\CurrentControlSet\Services\AFD\Parameters

The `EnableDynamicBacklog` registry value is a global switch to enable or disable a dynamic backlog. Setting it to 1 enables the dynamic backlog queue.

`MinimumDynamicBacklog` controls the minimum number of free connections allowed on a single TCP port. If the number of free connections drops below this value, then additional free connections are created automatically. Recommended value is 20.

The `MaximumDynamicBacklog` registry value defines the sum of active half-open connections and the maximum number of free connections. When this value is exceeded, no more free connections will be created by a system. Microsoft suggests that this value should not exceed 20000.

The last `DynamicBacklogGrowthDelta` parameter controls the number of free connections to be created when additional connections are necessary. Recommended value: 10.

The table below shows the recommended values for the AFD.SYS driver:

Subkey Registry Value Entry	Format	Value
EnableDynamicBacklog	DWORD	1
MinimumDynamicBacklog	DWORD	20
MaximumDynamicBacklog	DWORD	20000
DynamicBacklogGrowthDelta	DWORD	10

### Operating system: Linux

A `tcp_max_syn_backlog` variable defines how many half-open connections can be kept by the backlog queue. For instance 256 is a total number of half-open connections handled in memory by Linux RedHat 7.3. The TCP/IP stack variables can be configured by `sysctl` or standard Unix commands. The following example shows how to change the default size of the backlog queue by the `sysctl` command:

```
# sysctl -w net.ipv4.tcp_max_syn_backlog="2048"
```

## Operating system: Sun Solaris

In Sun Solaris there are two parameters which control the maximum number of connections. The first parameter controls the total number of full connections. The second `tcp_conn_req_max_q0` parameter defines how many half-open connections are allowed without the dropping of incoming requests. In Sun Solaris 8, the default value is set to 1024. Using the `ndd` command we can modify this value.

```
# ndd -set /dev/tcp tcp_conn_req_max_q0 2048
```

## Operating system: HP-UX

In HP-UX, a `tcp_syn_rcvd_max` TCP/IP stack variable is responsible for control of the maximum number of half-open connections in the SYN RECEIVE state. In HP-UX 11.00 this value is set to 500. We can change this value by using the `ndd` command, similar to the one used in a Sun Solaris system.

```
# ndd -set /dev/tcp tcp_syn_rcvd_max 2048
```

## Decreasing total time of handling connection request

As we know, SYN flooding/spoofing attacks are simply a series of SYN packets, mostly from forged IP addresses. In the last section we tried to increase the backlog queue. Now that our systems can handle more SYN requests, we should decrease the total time we keep half-open connections in the backlog queue. When a server receives a request, it immediately sends a response with the SYN and ACK flags set, puts this half-open connection into the backlog queue, and then waits for a packet with the ACK flag set from the client. When no response is received from the client, the server retransmits a response packet (with the SYN and ACK flags set) several times (depending on default value in each operating system) by giving the client a chance to send the ACK packet again. It is clear that when the source IP address of client was spoofed, the ACK packet will never arrive. After a few minutes the server removes this half-open connection. We can speed up this time of removing connections in the SYN RECEIVED state from the backlog queue by changing time of first retransmission and by changing the total number of retransmissions.

Another technique of protection against SYN attacks is switching off some TCP parameters that are always negotiated during the three-way handshake process. Some of these parameters are

automatically turned off by mechanisms described in the first section (`SynAttackProtect` and `Syncookies`).

Now, I will describe TCP/IP stack variables which allow a decrease in the time half-open connections are kept in the backlog queue.

### Operating system: Windows 2000

In Windows 2000, the default time for a first retransmission is set to 3 seconds (3000 milliseconds) and can be changed by modifying the value of the `TcpInitialRtt` registry entry (for every interface). For example, to decrease time of a first retransmission to 2 seconds we have to set this registry value to 2000 milliseconds in decimal format. The number of retransmissions (packets with the SYN and ACK flags set) is controlled by a `TcpMaxConnectResponseRetransmissions` registry parameter which has to be added to `HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters` registry key.

The table below contains a few examples of values and corresponding times for keeping half-open connections in the backlog queue (the time of a first retransmission is set to 3 seconds).

Value	Time of retransmission	Total time to keep half-open connections in the backlog queue
1	in 3rd second	9 seconds
2	in 3rd and 9th second	21 seconds
3	in 3rd , 9th and 21st second	45 seconds

We can set this registry value to 0, whereby Windows doesn't try to retransmit packets at all. In this case, the system sends only one response and cancels the half-open connection after 3 seconds. This setting is ignored when its value is equal or greater than 2 and when `SynAttackProtect` is enabled.

### Operating system: Linux RedHat

A `tcp_synack_retries` variable is responsible for controlling the number of retransmissions in

Linux operating system. Its default value is set to 5 for most Linux operating systems, which causes the half-open connection to be removed after 3 minutes. In the below table there are calculations for other values.

<b>Value</b>	<b>Time of retransmission</b>	<b>Total time to keep half-open connections in the backlog queue</b>
1	in 3rd second	9 seconds
2	in 3rd and 9th second	21 seconds
3	in 3rd , 9th and 21st second	45 seconds

### **Operating system: Sun Solaris**

In this operating system it is impossible to turn off retransmissions of packets directly using the `ndd` command. Moreover, in Sun Solaris there are parameters which are non-configurable by `ndd` and which control the number of retransmissions (at least 3) and total time of packet retransmissions (at least 3 minutes). More information about these parameters can be found in the "[Solaris 2.x - Tuning Your TCP/IP stack and More](#)" document.

### **Operating system: HP-UX**

For HP-UX, the time spent handling half-open connections in the backlog queue is controlled by the `tcp_ip_abort_cinterval` parameter. By using the `ndd` command we can define how long a HP-UX operating system will be waiting for the ACK packet. We can control how many retransmissions will be performed indirectly by changing this value. Have a look at the table below.

<b>Value</b>	<b>Time of retransmission</b>	<b>Total time to keep half-open connections in the backlog queue</b>
1000	-	1 second

5000	in 2nd second	5 seconds
10000	in 2nd and 5th second	10 seconds
60000	In 2nd, 5th, 11th, 23rd and 47th second	1 minute

We can change the time of a first retransmission by modifying `tcp_rexmit_interval_initial`. Intervals of subsequent retransmissions are controlled by two parameters: `tcp_rexmit_interval` and `tcp_rexmit_interval_min`. These three variables are the same as in a Sun Solaris operating system.

## Summary

The methods of hardening the TCP/IP stack that are presented in this article make servers more resistant to SYN flooding and SYN spoofing - Denial of Service attacks. A modification of your default TCP/IP stack settings is also recommended during the process of securing of the operating system.

---

*Mariusz Burdach is a computer security consultant who specializes in vulnerability assessment, intrusion detection and computer forensics. During the last few years he has worked as a consultant in the European Network Security Institute where he conducted penetration tests, vulnerability assessments and security audits for Internet banks, government and financial institutions in Poland. He is co-author of the Solaris Security Administrator's Guide, a step-by-step guide to securing SUN's Solaris operating system. Comments on this article are appreciated, send them to [M\\_Burdach@comfort.pl](mailto:M_Burdach@comfort.pl).*

## References

- ["Microsoft Windows 2000 TCP/IP Implementation Details"](#)
- ["How To: Harden the TCP/IP Stack Against Denial of Service Attacks in Windows 2000"](#)
- ["How To: Harden the TCP/IP Stack Against Denial of Service Attacks in Windows Server 2003"](#)
- ["Solaris\[tm\] Operating Environment Network Settings for Security"](#)
- ["Building a bastion host using hp-ux 11"](#)
- ["Solaris 2.x - Tuning Your TCP/IP Stack and More"](#)
- [SYN cookies mechanism](#)
- [Phrack Magazine 48 "Project Neptune"](#)
- ["G.E.N.E.S.I.S"](#)
- ["Distributed Reflection Denial of Service"](#)

[Privacy Statement](#)

Copyright 2006, SecurityFocus