Internet Engineering Task Force
INTERNET DRAFT
draft-floyd-tcp-reset-04.txt
May, 2002
Expires: November, 2002

Sally Floyd
ICIR

Inappropriate TCP Resets Considered Harmful

Status of this Memo

Abstract

This document is being written because there are a number of firewalls in the Internet that inappropriately reset a TCP connection upon receiving certain TCP SYN packets, in particular, packets with flags set in the Reserved field of the TCP header. In this document we argue that this practice is not conformant with TCP standards, and is an inappropriate overloading of the semantics of the TCP reset. We also consider the longer-term consequences of this and similar actions as obstacles to the evolution of the Internet infrastructure.

1. Introduction

TCP uses the RST (Reset) bit in the TCP header to reset a TCP connection. Resets are appropriately sent in response to a connection request to a nonexistent connection, for example. The TCP receiver of the reset aborts the TCP connection, and notifies the application [RFC793, RFC1122, Ste94].

Unfortunately, a number of firewalls and load-balancers in the current Internet send a reset in response to a TCP SYN packet that use flags from the Reserved field in the TCP header. Section 3 below discusses the specific example of firewalls that send resets in response to TCP SYN packets from ECN-capable hosts.

This document is being written to inform administrators of web servers and firewalls of this problem, in an effort to encourage the deployment of bug-fixes [FIXES]. A second purpose of this document is to consider the longer-term consequences of such middlebox behavior on the more general evolution of protocols in the Internet.

2. The history of TCP resets.

This section gives a brief history of the use of the TCP reset in the TCP standards, and argues that sending a reset in response to a SYN packet that uses bits from the Reserved field of the TCP header is non-compliant behavior.

RFC 793 contained the original specification of TCP in September, 1981 [RFC793]. This document defined the RST bit in the TCP header, and explained that reset was devised to prevent old duplicate connection initiations from causing confusion in TCP's three-way handshake. The reset is also used when a host receives data for a TCP connection that no longer exists.

RFC 793 states the following, in Section 5:

"As a general rule, reset (RST) must be sent whenever a segment arrives which apparently is not intended for the current connection. A reset must not be sent if it is not clear that this is the case."

RFC 1122 "amends, corrects, and supplements" RFC 793. RFC 1122 says nothing specific about sending resets, or not sending resets, in response to flags in the TCP Reserved field.

Thus, there is nothing in RFC 793 or RFC 1122 that suggests that it is acceptable to send a reset simply because a SYN packet uses Reserved flags in the TCP header, and RFC 793 explicitly forbids sending a reset for this reason.

RFC 793 and RFC 1122 both include Jon Postel's famous robustness principle, also from RFC 791: "Be liberal in what you accept, and conservative in what you send." RFC 1122 reiterates that this robustness principle "is particularly important in the Internet layer, where one misbehaving host can deny Internet service to many other hosts." The discussion of the robustness principle in RFC 1122 also states that "adaptability to change must be designed into all levels of Internet host software". The principle "be liberal in what you accept" doesn't carry over in a clear way (if at all) to the world of firewalls, but the issue of "adaptability to change" is crucial nevertheless. The challenge is to protect legitimate security interests without completely blocking the ability of the Internet to evolve to support new applications, protocols, and functionality.

2.1. The TCP Reserved Field

RFC 793 says that the Reserved field in the TCP header is reserved for future use, and must be zero. A rephrasing more consistent with the rest of the document would have been to say that the Reserved field should be zero when sent and ignored when received, unless specified otherwise by future standards actions. However, the phrasing in RFC 793 does not permit sending resets in response to TCP packets with a non-zero Reserved field, as is explained in the section above.

2.2. Behavior of and Requirements for Internet Firewalls

RFC 2979 on the Behavior of and Requirements for Internet Firewalls [RFC2979], an Informational RFC, contains the following:

"Applications have to continue to work properly in the presence of firewalls. This translates into the following transparency rule: The introduction of a firewall and any associated tunneling or access negotiation facilities MUST NOT cause unintended failures of legitimate and standards-compliant usage that would work were the firewall not present."

"A necessary corollary to this requirement is that when such failures do occur it is incumbent on the firewall and associated software to address the problem: Changes to either implementations of existing standard protocols or the protocols themselves MUST NOT be necessary."

"Note that this requirement only applies to legitimate protocol usage and gratuitous failures -- a firewall is entitled to block any sort of access that a site deems illegitimate, regardless of whether or not the attempted access is standards-compliant."

We would note that RFC 2979 is an Informational RFC. RFC 2026 on Internet Standards Process says the following in Section 4.2.2: "An 'Informational' specification is published for the general information of the Internet community, and does not represent an Internet community consensus or recommendation" [RFC2026].

2.3. Sending Resets as a Congestion Control Mechanism

Some firewalls and hosts send resets in response to SYN packets as a congestion control mechanism, for example, when their listen queues are full. These resets are sent without regard to the contents of the TCP Reserved field. Possibly in response to the use of resets as a congestion control mechanism, several popular TCP implementations immediately resend a SYN packet in response to a reset, up to four times.

We would recommend that the TCP reset not be used as a congestion control mechanism, because this overloads the semantics of the reset message, and inevitably leads to more aggressive behavior from TCP implementations in response to a reset. We would suggest that simply dropping the SYN packet is the most effective response to congestion. The TCP sender will retransmit the SYN packet, using the default value for the Retransmission Timeout (RTO), backing-off the retransmit timer after each retransmit.

2.4. Resets in Response to Changes in the Precedence Field

RFC 793 includes the following in Section 5:

"If an incoming segment has a security level, or compartment, or precedence which does not exactly match the level, and compartment, and precedence requested for the connection, a reset is sent and connection goes to the CLOSED state."

The "precedence" refers to the (old) Precedence field in the (old) ToS field in the IP header. The "security" and "compartment" refer to the obsolete IP Security option. When it was written, this was consistent with the guideline elsewhere in RFC 793 that resets should only be sent when a segment arrives which apparently is not intended for the current connection.

RFC 2873 on "TCP Processing of the IPv4 Precedence Field" discusses specific problems raised by the sending of resets when the precedence field has changed [RFC2873]. RFC 2873, currently a Proposed Standard, specifies that TCP must ignore the precedence of all received segments, and must not send a reset in response to changes in the precedence field. We discuss this here to clarify that this issue never permitted the sending of a reset in response to a segment with a non-zero TCP Reserved field.

2.5. Resets in Response to Illegal Option Lengths

RFC 1122 says the following in Section 4.2.2.5 about TCP options [RFC1122]:

"A TCP MUST be able to receive a TCP option in any segment. A TCP MUST ignore without error any TCP option it does not implement, assuming that the option has a length field (all TCP options defined in the future will have length fields). TCP MUST be prepared to handle an illegal option length (e.g., zero) without crashing; a suggested procedure is to reset the connection and log the reason."

This makes sense, as a TCP receiver is unable to interpret the rest of the data on a segment that has a TCP option with an illegal option length. Again, we discuss this here to clarify that this issue never permitted the sending of a reset in response to a segment with a non-zero TCP Reserved field.

3. The Specific Example of ECN

This section has a brief explanation of ECN (Explicit Congestion Notification) in general, and the ECN-setup SYN packet in particular.

The Internet is based on end-to-end congestion control, and historically the Internet has used packet drops as the only method for routers to indicate congestion to the end nodes. ECN is a recent addition to the IP architecture to allow routers to set a bit in the IP packet header to inform end-nodes of congestion, instead of dropping the packet. ECN requires the cooperation of the transport end-nodes.

The ECN specification, RFC 2481, was an Experimental RFC from January 1999 until June 2001, when a revised document [RFC3168] was approved as Proposed Standard. More information about ECN is available from the ECN Web Page [ECN].

The use of ECN with TCP requires that both TCP end-nodes have been upgraded to support the use of ECN, and that both end-nodes agree to use ECN with this particular TCP connection. This negotiation of ECN support between the two TCP end-nodes uses two flags that have been allocated from the Reserved field in the TCP header [RFC2481].

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |                       | U | A | P | R | S | F |
| Header Length |        Reserved       | R | C | S | S | Y | I |
|               |                       | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 1: The previous definition of bytes 13 and 14 of the TCP header.

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |               | C | E | U | A | P | R | S | F |
| Header Length |    Reserved   | W | C | R | C | S | S | Y | I |
|               |               | R | E | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 2: The current definition of bytes 13 and 14 of the TCP Header, from RFC 3168.

The two ECN flags in the TCP header are defined from the last two bits in the Reserved field of the TCP header. Bit 9 in the Reserved field of the TCP header is designated as the ECN-Echo flag (ECE), and Bit 8 is designated as the Congestion Window Reduced (CWR) flag. To negotiate ECN usage, the TCP sender sends an "ECN-setup SYN packet", a TCP SYN packet with the ECE and CWR flags set. If the TCP host at the other end wishes to use ECN for this connection, then it sends an "ECN-setup SYN-ACK packet", a TCP SYN packet with the ECE flag set and the CWR flag not set. Otherwise, the TCP host at the other end returns a SYN-ACK packet with neither the ECE nor the CWR flag set.

So now back to TCP resets. When a TCP host negotiating ECN sends an ECN-setup SYN packet, an old TCP implementation is expected to ignore those flags in the Reserved field, and to send a plain SYN-ACK packet in response. However, there are some broken firewalls and load-balancers in the Internet that instead respond to an ECN-setup SYN packet with a reset. Following the deployment of ECN-enabled end nodes, there were widespread complaints that ECN-capable hosts could not access a number of websites [Kelson00]. This has been investigated by the Linux community, and by the TBIT project [TBIT] in data taken from September, 2000, up to March, 2002, and has been discussed in an article in Enterprise Linux Today [Cou01]. Some of the offending equipment has been identified, and a web page [FIXES] contains a list of non-compliant products and the fixes posted by the vendors. In March 2002, six months after ECN was approved as Proposed Standard, ECN-setup SYN packets were answered by a reset from 203 of the 12,364 web sites tested, and ECN-setup SYN packets were dropped for 420 of the web sites. Installing software that blocks packets using flags in TCP's Reserved field is considerably easier than uninstalling that software later on.

3.1. ECN: The Work-Around.

A work-around for maintaining connectivity in the face of the broken equipment was described in [Floyd00], and has been specified in RFC 3168 as a procedure that may be included in TCP implementations. We describe this work-around briefly below.

To provide robust connectivity even in the presence of faulty equipment, a TCP host that receives a reset in response to the transmission of an ECN-setup SYN packet may resend the SYN with CWR and ECE cleared. This would result in a TCP connection being established without using ECN. This also has the unfortunate result of the ECN-capable TCP host not responding properly to the first valid reset. If a second reset is sent in response to the second SYN, which had CWR and ECE cleared, then the TCP host should respond properly by aborting the connection.

Similarly, a host that receives no reply to an ECN-setup SYN within the normal SYN retransmission timeout interval may resend the SYN and any subsequent SYN retransmissions with CWR and ECE cleared. To overcome normal packet loss that results in the original SYN being lost, the originating host may retransmit one or more ECN-setup SYN packets before giving up and retransmitting the SYN with the CWR and ECE bits cleared.

Some TCP implementors have so far decided not to deploy these workarounds, for the following reasons:

* The work-arounds would result in ECN-capable hosts not responding properly to the first valid reset received in response to a SYN packet.

* The work-arounds would limit ECN functionality in environments without broken equipment, by disabling ECN where the first SYN or SYN-ACK packet was dropped in the network.

* The work-arounds in many cases would involve a delay of six seconds or more before connectivity is established with the remote server, in the case of broken equipment that drops ECN-setup SYN packets. By accommodating this broken equipment, the work-arounds have been judged as implicitly accepting both this delay and the broken equipment that would be causing this delay.

One possibility would be for such work-arounds to be configurable by the user.

One unavoidable consequence of the work-around of resending a modified SYN packet in response to a reset is to further erode the semantics of the TCP reset. Thus, when a box sends a reset, the TCP host receiving that reset does not know if the reset was sent simply because of the ECN-related flags in the TCP header, or because of some more fundamental problem. Therefore, the TCP host resends the TCP SYN packet without the ECN-related flags in the TCP header. The ultimate consequence of this absence of clear communications from the middlebox to the end-nodes could be an extended spiral of communications specified for transport protocols, as end nodes attempt to sacrifice as little functionality as possible in the process of determining which packets will and will not be forwarded to the other end. This is discussed in more detail in Section 6.1 below.

4. On Combating Obstacles to the Proper Evolution of the Internet Infrastructure

One of the reasons that this issue of inappropriate resets is important (to me) is that it has complicated the deployment of ECN in the Internet (though it has fortunately not blocked the deployment completely). It has also added an unnecessary obstacle to the future effectiveness of ECN.

However, a second, more general reason why this issue is important is that the presence of equipment in the Internet that rejects valid TCP packets limits the future evolution of TCP, completely aside from the issue of ECN. That is, the widespread deployment of equipment that rejects TCP packets that use Reserved flags in the TCP header could effectively prevent the deployment of new mechanisms that use any of these Reserved flags. It doesn't matter if these new mechanisms have the protection of Experimental or Proposed Standard status from the IETF, because the broken equipment in the Internet does not stop to look up the current status of the protocols before rejecting the packets. TCP is good, and useful, but it would be a pity for the deployment of broken equipment in the Internet to result in the "freezing" of TCP in its current state, without the ability to use the Reserved flags in the future evolution of TCP.

In the specific case of middleboxes that block TCP SYN packets attempting to negotiate ECN, the work-around described in Section 3.1 is sufficient to ensure that end-nodes could still establish connectivity. However, there are likely to be additional uses of the TCP Reserved Field standardized in the next year or two, and these additional uses might not coexist quite as successfully with middleboxes that send resets. Consider the difficulties that could result if a path changes in the middle of a connection's lifetime, and the middleboxes on the old and new paths have different policies about exactly which flags in the TCP Reserved field they would and would not block.

Taking the wider view, the existence of web servers or firewalls that send inappropriate resets is only one example of functionality in the Internet that restricts the future evolution of the Internet. The impact of all of these small restrictions taken together presents a considerable obstacle to the development of the Internet architecture.

5. Issues for Transport Protocols

One lesson for designers of transport protocols is that transport protocols will have to protect themselves from the unknown and seemingly arbitrary actions of firewalls, normalizers, and other middleboxes in the network. For the moment, for TCP, this means sending a non-ECN-setup SYN when a reset is received in response to an ECN-setup SYN packet. Defensive actions on the side of transport protocols could include using Reserved flags in the SYN packet before using them in data traffic, to protect against middleboxes that block packets using those flags. It is possible that transport protocols will also have to add additional checks during the course of the connection lifetime to check for interference from middleboxes along the path.

The ECN standards document, RFC 3168, contains an extensive discussion in Section 18 on "Possible Changes to the ECN Field in the Network", but includes the following about possible changes to the TCP header:

"This document does not consider potential dangers introduced by changes in the transport header within the network. We note that when IPsec is used, the transport header is protected both in tunnel and transport modes [ESP, AH]."

With the current modification of transport-level headers in the network by firewalls (as discussed below in Section 6.2), future protocol designers might no longer have the luxury of ignoring the possible impact of changes to the transport header within the network.

Transport protocols will also have to respond in some fashion to an ICMP code of "Communication Administratively Prohibited" if middleboxes start to use this form of the ICMP Destination Unreachable message to indicate that the packet is using functionality not allowed [RFC1812].

6. Issues for Middleboxes

Given that some middleboxes are going to drop some packets because they use functionality not allowed by the middlebox, the larger issue remains of how middleboxes should communicate the reason for this action to the end-nodes, if at all. One suggestion, for consideration in more depth in a separate document, would be that firewalls send an ICMP Destination Unreachable message with the code "Communication Administratively Prohibited" [B01].

We acknowledge that this is not an ideal solution, for several reasons. First, middleboxes along the reverse path might block these ICMP messages. Second, some firewall operators object to explicit communication because it reveals too much information about security policies. And third, the response of transport protocols to such an ICMP message is not yet specified.

However, an ICMP "Administratively Prohibited" message could be a reasonable addition, for firewalls willing to use explicit communication. One possibility, again to be explored in a separate document, would be for the ICMP "Administratively Prohibited" message to be modified to convey additional information to the end host.

We would note that this document does not consider middleboxes that block complete transport protocols. We also note that this document is not addressing firewalls that send resets in response to a TCP SYN packet to a firewalled-off TCP port. Such a use of resets seems consistent with the semantics of TCP reset. This document is only considering the problems caused by middleboxes that block specific packets within a transport protocol when other packets from that transport protocol are forwarded by the middlebox unaltered.

One complication is that once a mechanism is installed in a firewall to block a particular functionality, it can take considerable effort for network administrators to "un-install" that block. It has been suggested that tweakable settings on firewalls could make recovery from future incidents less painful all around. Again, because this document does not address more general issues about firewalls, the issue of greater firewall flexibility, and the attendant possible security risks, belongs in a separate document.

6.1. Current Choices for Firewalls

Given a firewall that has decided to drop TCP packets that use reserved bits in the TCP header, one question is whether the firewall should also send a Reset, in order to prevent the TCP connection from consuming unnecessary resources at the TCP sender waiting for the retransmit timeout. We would argue that whether or not the firewall feels compelled to drop the TCP packet, it is not appropriate to send a TCP reset. Sending a TCP reset in response to prohibited functionality would continue the current overloading of the semantics of the TCP reset in a way that could be counterproductive all around.

As an example, Section 2.3 has already observed that some firewalls send resets in response to TCP SYN packets as a congestion control mechanism. Possibly in response to this (or perhaps in response to something else), some popular TCP implementations immediately resend a SYN packet in response to a reset, up to four times. Other TCP implementations, in conformance to the standards, don't resend SYN packets after receiving a reset. The more aggressive TCP implementations increase congestion for others, but also increase their own chances of eventually getting through. Giving these fluid semantics for the TCP reset, one might expect more TCP implementations to start resending SYN packets in response to a reset, completely apart from any issues having to do with ECN. Obviously, this weakens the effectiveness of the reset when used for its original purpose, of responding to TCP packets that apparently are not intended for the current connection.

If we add to this mix the use of the TCP reset by firewalls in response to TCP packets using reserved bits in the TCP header, this muddies the waters further. Because TCP resets could be sent due to congestion, or to prohibited functionality, or because a packet was received from a previous TCP connection, TCP implementations (or, more properly, TCP implementors) would now have an incentive to be even more persistent in resending SYN packets in response to TCP resets. In addition to the incentive mentioned above of resending TCP SYN packets to increase one's odds of eventually getting through in a time of congestion, the TCP reset might have been due to prohibited functionality instead of congestion, so the TCP implementation might resend SYN packets in different forms to determine exactly which functionality is being prohibited. Such a continual changing of the semantics of the TCP reset could be expected to lead to a continued escalation of measures and countermeasures between firewalls and end-hosts, with little productive benefit to either side.

It could be argued that *dropping* the TCP SYN packet due to the use of prohibited functionality leads to overloading of the semantics of a packet drop, in the same way that the reset leads to overloading the semantics of a reset. This is true; from the viewpoint of end-system response to messages with overloaded semantics, it would be preferable to have an explicit indication about prohibited functionality (for those firewalls for some reason willing to use explicit indications). But given a firewall's choice between sending a reset or just dropping the packet, we would argue that just dropping the packet does less damage, in terms of giving an incentive to end-hosts to adopt counter-measures. It is true that just dropping the packet, without sending a reset, results in delay for the TCP connection in resending the SYN packet without the prohibited functionality. However, sending a reset has the undesirable longer-term effect of giving an incentive to future TCP implementations to add more baroque combinations of resending SYN packets in response to a reset, because the TCP sender can't tell if the reset is for a standard reason, for congestion, or for the prohibited functionality of option X or reserved bit Y in the TCP header.

6.2. The Complications of Modifying Packet Headers in the Network

In addition to firewalls that send resets in response to ECN-setup SYN packets and firewalls that drop ECN-setup SYN packets, there also exist firewalls that by default zero the flags in the TCP Reserved field, including the two flags used for ECN. We note that in some cases this could have unintended and undesirable consequences.

If a firewall zeros the ECN-related flags in the TCP header in the initial SYN packet, then the TCP connection will be set up without using ECN, and the ECN-related flags in the TCP header will be sent zeroed-out in all of the subsequent packets in this connection. This will accomplish the firewall's purpose of blocking ECN, while allowing the TCP connection to proceed efficiently and smoothly without using ECN.

If for some reason the ECN-related flags in the TCP header aren't zeroed in the initial SYN packet from host A to host B, but the firewall does zero those flags in the responding SYN/ACK packet from host B to host A, the consequence could be to subvert end-to-end congestion control for this connection. The ECN specifications were not written to ensure robust operation in the presence of the arbitrary zeroing of TCP header fields within the network, because it didn't occur to the authors of the protocol at the time that this was a requirement in protocol design.

Similarly, if the ECN-related flags in the TCP header are not zeroed in either the SYN or the SYN/ACK packet, but the firewall does zero these flags in later packets in that TCP connection, this could also have the unintended consequence of subverting end-to-end congestion control for this connection. The details of these possible interactions are not crucial for this document, and are described in the appendix. However, our conclusion, both for the ECN-related flags in the TCP header and for future uses of the four other bits in the TCP Reserved field, would be that if it is required for firewalls to be able to block the use of a new function being added to a protocol, this is best addressed in the initial design phase by joint cooperation between the firewall community and the protocol designers.

7. Conclusions

Our conclusion is that it is not conformant with current standards for a firewall, load-balancer, or web-server to respond with a reset to a TCP SYN packet simply because the packet uses flags in the TCP Reserved field. More specifically, it is not conformant to respond with a reset to a TCP SYN packet simply because the ECE and CWR flags are set in the IP header. We would urge vendors to make available fixes for any nonconformant code, and we could urge ISPs and system administrators to deploy these fixes in their web servers and firewalls.

We don’t claim that it violates any standard for middleboxes to arbitrarily drop packets that use flags in the TCP Reserved field, but we would argue that behavior of this kind, without a clear method for informing the end-nodes of the reasons for these actions, could present a significant obstacle to the development of TCP. More work is clearly needed to reconcile the conflicting interests of providing security while at the same time allowing the careful evolution of Internet protocols.

8. Acknowledgements

This document results from discussions and activity by many people, so I will refrain from trying to acknowledge all of them here. My specific thanks go to Ran Atkinson, Steve Bellovin, Alex Cannara, Dennis Ferguson, Ned Freed, Mark Handley, John Klensin, Allison Mankin, Jitendra Padhye, Vern Paxson, K. K. Ramakrishnan, Jamal Hadi Salim, Pekka Savola, Alex Snoeren, and Dan Wing for feedback on this document, and to the End-to-End Research Group, the IAB, and the IESG for discussion of these issues. I thank Mikael Olsson for numerous rounds of feedback. I also thank the members of the Firewall Wizards mailing list for feedback (generally of disagreement) on an earlier draft of this document.

Email discussions with a number of people, including Dax Kelson, Alexey Kuznetsov, Kacheong Poon, David Reed, Jamal Hadi-Salim, and Venkat Venkatsubra, have addressed the issues raised by non-conformant equipment in the Internet that does not respond to TCP SYN packets with the ECE and CWR flags set. We thank Mark Handley, Jitentra Padhye, and others for discussions on the TCP initialization procedures.

9. Normative References

[RFC793] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", RFC 793, DARPA, September 1981.

[RFC1122] Braden, R., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1812] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC2026] S. Bradner, The Internet Standards Process -- Revision 3, RFC 2026, BCP 9, October 1996.

[RFC2481] K. K. Ramakrishnan and S. Floyd, A Proposal to add Explicit Congestion Notification (ECN) to IP, RFC 2481, January 1999.

[RFC2873] Xiao, X., Hannan, A., Paxson, V., and Crabbe, E., TCP Processing of the IPv4 Precedence Field, RFC 2873, June 2000. Proposed Standard.

[RFC2979] N. Freed, Behavior of and Requirements for Internet Firewalls, RFC 2979, Informational, October 2000.

[RFC3168] K.K. Ramakrishnan, S. Floyd, and D. Black, The Addition of Explicit Congestion Notification (ECN) to IP, RFC 3168, Proposed Standard, September 2001.

10. Informative References

[B01] Steve Bellovin, A "Reason" Field for ICMP "Administratively Prohibited" Messages, draft-bellovin-adminprohib-00.txt, internet-draft, work in progress, December 2001.

[Cou01] Scott Courtney, Why Can't My 2.4 Kernel See Some Web Sites?, Enterprise Linux Today, Apr 17, 2001. URL "http://eltoday.com/article.php3?ltsn=2001-04-17-001-14-PS".

[ECN] "The ECN Web Page", URL "http://www.icir.org/floyd/ecn.html".

[FIXES] ECN-under-Linux Unofficial Vendor Support Page, URL "http://gtf.org/garzik/ecn/".

[Floyd00] Sally Floyd, Negotiating ECN-Capability in a TCP connection, October 2, 2000, email to the end2end-interest mailing list. URL "http://www.icir.org/floyd/papers/ECN.Oct2000.txt".

[Kelson00] Dax Kelson, note sent to the Linux kernel mailing list, September 10, 2000.

[QUESO] Toby Miller, Intrusion Detection Level Analysis of Nmap and Queso, August 30, 2000. URL "http://www.securityfocus.com/infocus/1225".

[Ste94] Stevens, W., "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, 1994.

[SFO01] FreeBSD ipfw Filtering Evasion Vulnerability, Security Focus Online, January 23, 2001. URL "http://www.securityfocus.com/bid/2293".

[TBIT] Jitendra Padhye and Sally Floyd, Identifying the TCP Behavior of Web Servers, SIGCOMM, August 2001. URL "http://www.icir.org/tbit/".

11. Security Considerations

One general risk of using Reserved flags in TCP is the risk of providing additional information about the configuration of the host in question. However, TCP is sufficiently loosely specified as it is, with sufficiently many variants and options, that port-scanning tools such as Nmap and Queso do rather well in identifying the configuration of hosts even without the use of Reserved flags.

The security considerations and all other considerations of a possible ICMP Destination Unreachable message with the code "Communication Administratively Prohibited" will be discussed in a separate document.

The traditional concern of firewalls is to prevent unauthorized access to systems, to prevent DoS attacks and other attacks from subverting the end-user terminal, and to protect end systems from buggy code. We are aware of one security vulnerability reported from the use of the Reserved flags in the TCP header [SFO01]. A packet filter intended only to let through packets in established connections can let pass a packet not in an established connection if the packet has the ECE flag set in the reserved field. "Exploitation of this vulnerability may allow for unauthorized remote access to otherwise protected services." It is also possible that an implementation of TCP could appear that has buggy code associated with the use of Reserved flags in the TCP header, but we are not aware of any such implementation at the moment.

Unfortunately, misconceived security concerns are one of the reasons for the problems described in this document in the first place. An August, 2000, article on "Intrusion Detection Level Analysis of Nmap and Queso" described the port-scanning tool Queso as sending SYN packets with the last two Reserved bits in the TCP header set, and said the following: "[QUESO] is easy to identify, if you see [these two Reserved bits and the SYN bit] set in the 13th byte of the TCP header, you know that someone has malicious intentions for your network." As is documented on the TBIT Web Page, the middleboxes that block SYNs using the two ECN-related Reserved flags in the TCP header do not block SYNs using other Reserved flags in the TCP header.

One lesson appears to be that anyone can effectively ''attack'' a new TCP function simply by using that function in their publicly-available port-scanning tool, thus causing middleboxes of all kinds to block the use of that function.

12. Appendix: The Complications of Modifying Packet Headers

In this section we first show that if the ECN-related flags in the TCP header aren't zeroed in the initial SYN packet from Host A to Host B, but are zeroed in the responding SYN/ACK packet from Host B to Host A, the consequence could be to subvert end-to-end congestion control for this connection.

Assume that the ECN-setup SYN packet from Host A is received by Host B, but the ECN-setup SYN/ACK from Host B is modified by a firewall in the network to a non-ECN-setup SYN/ACK, as in Figure 3 below. RFC 3168 does not specify that the ACK packet in any way should echo the TCP flags received in the SYN/ACK packet, because it had not occurred to the designers that these flags would be modified within the network.

```
  Host A                     Firewall or router            Host B
  -----------------------------------------------------------------
  Sends ECN-setup SYN     ---------------->  Receives ECN-setup SYN
                                          <- Sends ECN-setup SYN/ACK
               <- Firewall zeros flags
  Receives non-ECN-setup SYN/ACK
  Sends ACK and data      ---------------->   Receives ACK and data
                                       <- Sends data packet with ECT
                     <- Router sets CE
  Receives data packet with ECT and CE
```

Figure 3: ECN-related flags in SYN/ACK packet cleared in network.

Following RFC 3168, Host A has received a non-ECN-setup SYN/ACK packet, and must not set ECT on data packets. Host B, however, does not know that Host A has received a non-ECN-setup SYN/ACK packet, and Host B may set ECT on data packets. RFC 3168 does not require Host A to respond properly to data packets received from Host B with the ECT and CE codepoints set in the IP header. Thus, the data sender, Host B, might never be informed about the congestion encountered in the network, thus violating end-to-end congestion control.

Next we show that if the ECN-related flags in the TCP header are not zeroed in either the SYN or the SYN/ACK packet, but the firewall does zero these flags in later packets in that TCP connection, this could also have the unintended consequence of subverting end-to-end congestion control for this connection. Figure 4 shows this scenario.

```
  Host A                    Firewall or router           Host B
  -----------------------------------------------------------------
  Sends ECN-setup SYN     ----------------->  Receives ECN-setup SYN
  Receives ECN-setup SYN/ACK <-------------  Sends ECN-setup SYN/ACK
  Sends ACK and data      ----------------->   Receives ACK and data
                                      <- Sends data packet with ECT
                     <- Router sets CE
  Receives data packet with ECT and CE
  Sends ACK with ECE ->
                        Firewall resets ECE ->
                                                   Receives plain ACK
```

Figure 4: ECN-related flags in ACK packet cleared in network.

The ECN-related flags are not changed by the network in the ECN-setup SYN and SYN/ACK packets for the scenario in Figure 4, and both end nodes are free to use ECN, and to set the ECT flag in the ECN field in the IP header. However, if the firewall clears the ECE flag in the TCP header in ACK packets from Node A to Node B, then Node B will never hear about the congestion that its earlier data packets encountered in the network, thus subverting end-to-end congestion control for this connection.

Additional complications will arise when/if the use of the ECN nonce in TCP becomes standardized in the IETF [RFC3168], as this could involve the specification of an additional flag from the TCP Reserved field for feedback from the TCP data receiver to the TCP data sender. The primary motivation for the ECN nonce is to allow mechanisms for the data sender to verify that network elements are not erasing the CE codepoint, and that data receivers are properly reporting to the sender the receipt of packets with the CE codepoint set.

13. IANA Considerations

There are no IANA considerations in this document.

AUTHORS' ADDRESSES

Sally Floyd
Phone: +1 (510) 666-2989
ICIR (ICSI Center for Internet Research)
Email: floyd@icir.org
URL: http://www.icir.org/floyd/

This draft was created in May 2002.
It expires November 2002.