Resilience of Deployed TCP to Blind FIN Attacks

Matthew Luckie

October 16, 2017

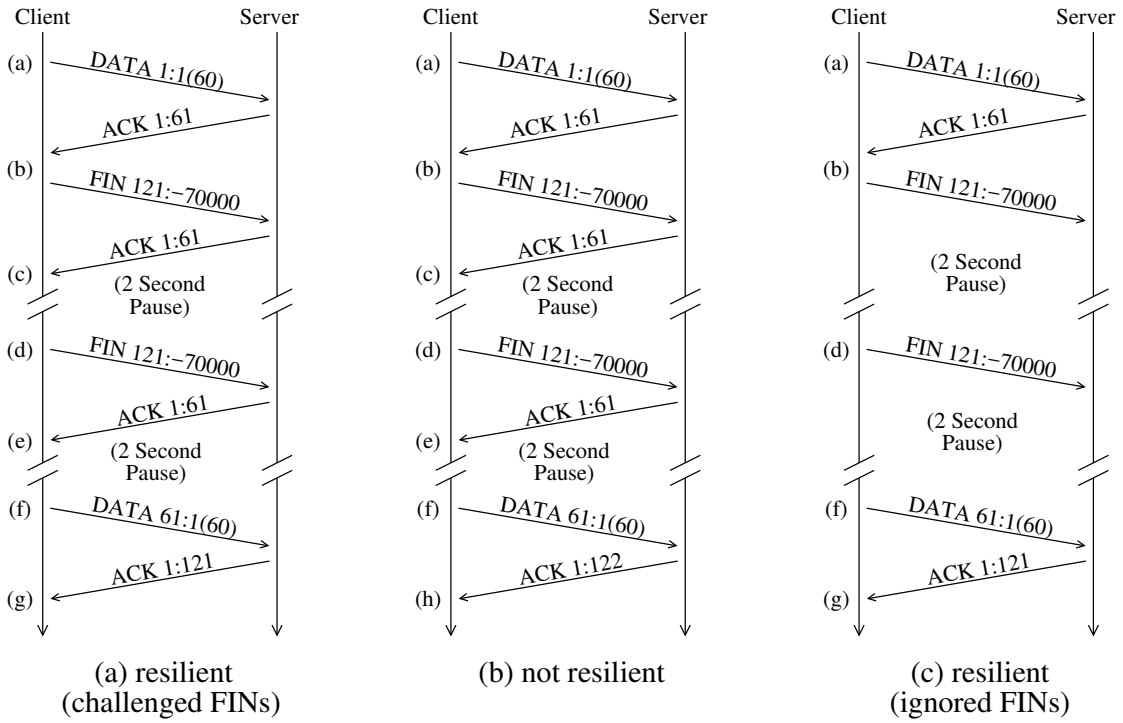
1 Introduction

In prior work we conducted in 2015 [3], we considered the resilience of deployed TCP implentations to blind in-window RST, SYN, and data attacks. These three attacks and defenses to the attacks were previously described in RFC5961 [4]. In this report, we consider the resilience of deployed TCP implementations to blind in-window FIN attacks, an attack not explicitly covered in RFC5961, where an off-path adversary disrupts an established connection by sending a packet that the victim believes came from its peer, causing the connection to be prematurely closed. We extended scamper [2], a parallelized packet prober with existing TCP behaviour inference capability, to add an active measurement test that infers whether or not a TCP implementation will accept a FIN packet that contains an acknowledgement value that should cause the receiver to discard the packet. We tested operating systems (and middleboxes deployed in front) of 4397 webservers in the wild in September 2017 and found 18% of tested connections were vulnerable to in-window FIN attack packets, consistent with our prior measurements testing the resilience of TCP implementations to blind in-window RST, SYN, and data attacks.

2 Method

Our method to test the resilience of deployed TCP to blind FIN attacks is similar to the method we used in prior work [3] to test response to blind data packets. We adopted an oracle-based approach by simulating a blind FIN attack on a TCP connection that we established. We broke the first segment of data (the HTTP GET or the first packet in the TLS handshake) into two pieces, and then sent the following sequence: (1) we sent the first piece with an acknowledgment number expected by the receiver, (2) we sent a FIN packet with a sequence number ahead of the receiver's *rcv.nxt* (leaving a hole for the second piece) and an acknowledgment number outside of the acceptable range defined by RFC 5961, and (3) we sent the second piece with an acknowledgment number expected by the receiver. We sent the FIN a second time, with a two second delay, to account for packet loss. If the receiver accepted the FIN packet with an invalid acknowledgement number, the second piece will fill a hole and the receiver will send an acknowledgement for the FIN.

Figure 1 illustrates our approach with three primary test outcomes. First, as in (a) in figure 1, the server might send a challenge ACK in response to the FIN packets but otherwise discard the FIN, as the ACK the server eventually sends after receiving the second piece of data does not also cover the FIN. This outcome indicates the server is not vulnerable to FIN packets that could



(a) resilient (challenged FINs)

(b) not resilient

(c) resilient (ignored FINs)

Figure 1: Overview of our blind FIN test. We break the first segment of data for the connection into three pieces. After the TCP handshake, we send the first piece (a) with an expected acknowledgment number, and then send the FIN (b) with an invalid acknowledgment number. We send the FIN twice to account for potential packet loss (b, d), and then the second piece (f) with an expected acknowledgment number. If the server rejected the FIN, then the ACK we receive will be just for the data contained in (f), as in (g). If the server accepted the FIN, then the ACK we receive will be for both the data contained in (f) and the FIN (b, d), as in (h). If the server ignored the FIN, then the ACK we receive will be just for the data contained in (f), as in (g), and we will not receive ACKs in response to the FINs we send.

have come from a blind attacker. Second, as in (b) in figure 1, the server might ACK the FIN packet as well as the second piece of data. This outcome indicates the server is vulnerable to FIN packets that could have come from a blind attacker. Third, as in (c) in figure 1, the server might not send any challenge ACKs in response to the FIN packets, and ignore the FIN, as the ACK the server eventually sends after receiving the second piece of data does not also cover the FIN. This outcome indicates the server is not vulnerable to FIN packets that could have come from a blind attacker. Finally, not shown in figure 1, the server might reset the connection in response to the FIN packet, which we detect when it sends a RST packet with a sequence number matching the incorrect acknowledgement value we set in our FIN packet.

3 Data

To test commodity TCP stacks as deployed in the wild, we established connections to websites in the Alexa list [1], and observed the behavior of their TCP implementations in response to our

Result	Frequency	Fraction
Accepted FIN	792	18.0%
Reset (ack-blind)	32	0.7%
Vulnerable	824	18.7%
Ignored (not vulnerable)	1205	27.4%
Challenge ACK (not vulnerable)	2283	51.9%
Not Vulnerable	3488	79.3%
Early FIN	8	0.2%
Early RST	20	0.5%
No connection	57	1.3%
Other	85	1.9%
Total	4397	100%

Table 1: Overview of results for the webserver population testing from cld-us VP in September 2015. 18.0% of the tested population incorrectly accepted the FIN packet, and 0.7% incorrectly reset the connection, instead of ignoring or challenging the FIN packet.

oracle's probing packets. We do not claim that our results are representative of any particular population; we tested websites to understand properties of currently deployed web operating systems and middleboxes. Nevertheless, we consider our results indicative of current likely behavior in other populations, including systems that support long-lived protocols including ssh and rsync.

From the cld-us Archiplego measurement vantage point on San Diego, we chose 5K random websites from the Alexa list. We sent a DNS query for each web site and used the first returned IPv4 address for the duration of our probing. Further, if multiple sites shared an IP address, we only probed the IP address once. This process yielded 4397 target IP addresses.

We then used the websites and target IP addresses collected to test those systems for resilience to FIN packets that could have come from a blind attacker.

4 Results

Table 1 summarizes the results for the webserver population we tested. We were able to test 98.1% of the webservers; we could not establish a TCP connection to other webservers, or they sent an early RST before we could complete our test. The overall results are encouraging as most (79.3%) did not acknowledge the FIN packet. Nevertheless, 18.7% of the tested population were vulnerable to FIN packets that could have come from a blinded attacker.

5 Code Release

We publicly release our code as part of the scamper package:
`http://www.caida.org/tools/measurement/scamper/`.

References

[1] Alexa. Top 1,000,000 sites. `http://www.alexa.com/topsites`.

[2] M. Luckie. Scamper: a scalable and extensible packet prober for active measurement of the internet. In *Proceedings of the 10th ACM SIGCOMM conference on Internet Measurement (IMC)*, pages 239–245, Nov. 2010.

[3] M. Luckie, R. Beverly, T. Wu, M. Allman, and k claffy. Resilience of deployed TCP to blind attacks. In *Proceedings of the 15th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 13–26, Oct. 2015.

[4] A. Ramaiah, R. Stewart, and M. Dalal. Improving TCP's robustness to blind in-window attacks. RFC 5961, Aug. 2010.