HAL open science

TCP Ack Storm DoS Attacks

Raz Abramov, Amir Herzberg

To cite this version:

Raz Abramov, Amir Herzberg. TCP Ack Storm DoS Attacks. 26th International Information Security Conference (SEC), Jun 2011, Lucerne, Switzerland. pp.29-40, 10.1007/978-3-642-21424-0_3 . hal-01567606

HAL Id: hal-01567606

https://inria.hal.science/hal-01567606v1

Submitted on 24 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

TCP Ack Storm DoS Attacks

Raz Abramov and Amir Herzberg

Bar Ilan University

Abstract. We present *Ack-storm DoS attacks*, a new family of DoS attacks exploiting a subtle design flaw in the core TCP specifications. The attacks can be launched by a very weak MitM attacker, which can only eavesdrop occasionally and spoof packets (a *Weakling in the Middle (WitM)*). The attacks can reach theoretically unlimited amplification; we measured amplification of over 400,000 against popular websites before aborting our trial attack.
Ack storm DoS attacks are practical. In fact, they are easy to deploy in large scale, especially considering the widespread availability of open wireless networks, allowing an attacker easy WitM abilities to thousands of connections. Storm attacks can be launched against the access network, e.g. blocking address to proxy web server, against web sites, or against the Internet backbone. Storm attacks work against TLS/SSL connections just as well as against unprotected TCP connections, but fails against IPsec or link-layer encrypted connections.
We show that Ack-storm DoS attacks can be easily prevented, by a simple fix to TCP, in either client or server, or using a packet-filtering firewall.

Keywords: Denial of service, TCP, secure network protocols.

1 Introduction

Most works in cryptography today adopt the all-powerful *Man In The Middle (MitM)* attacker model. The MitM attacker controls all of the traffic in the channels under him, with the ability to see, block and modify any package in the channel. In contrast, most works on Denial of Service (DoS) attacks, investigate the damage which much weaker attackers can cause, in order to focus on the most feasible and realistic attacks. Such weak attackers may only have the ability to send spoofed packets, or even weaker abilities - sending raw packets, sending only well-formed packets, or even merely issuing HTTP requests (e.g., puppets, see [5]).

In this work, we present and investigate the *Weakling In The Middle (WitM)* attacker model. The WitM attacker can eavesdrop on communication, but with significant limitations, mainly: eavesdropping only to one side of the connection, and receiving only a small percentage of the packets sent. These limitations are inspired by real-world wireless eavesdropping abilities, especially to open wireless networks; the 'one-sided' limitation is due to the fact that often the attacker is only able to eavesdrop to communication from the access point, and the low

percentage is due to the weak reception by a remote eavesdropper. Open wireless networks are becoming more and more common, whether its in restaurants, malls or even as a city-wide infrastructure [9]. Attackers today can eavesdrop on public networks from a distance without the need for special equipment, and with poor reception quality (capturing low percentage of packets). It is widely known how to make a directional('Yagi') antenna, that can reach up to 12 miles of range and cost no more than a few dollars (see [2] or numerous web pages).

In addition to their limited eavesdropping capabilities, WitM attackers can also send spoofed packets to the network. This ability is very common, since many ISPs fail to properly deploy Ingress filtering [1]. However, we restrict the number of packets that the attacker can send into the network per attack; real attackers will try to restrict the number of packets they send, in order to stay hidden and avoid capture. Note also that sending few packets per attack increases the number of attacks that the attacker can perform simultaneously. These aspects are similar to the stealth attacker model of [3].

We present several *'Ack-Storm DoS Attacks'* that, by injecting (two or more) packets into an existing TCP connection, cause a long exchange of TCP packets between a client and a server, terminated only by connection reset or packet losses. This way enables a WitM attacker to disrupt services to local and regional junctions in the Internet infrastructure, as well as well as to individual web sites and services.

The Ack-storm behavior of TCP has been mentioned before in [4] and [10] as a side effect of TCP hijacking attacks, and thus as something to be minimized and prevented.

We present a typical scenario in Fig. 1, with a client Alice connected to an open wifi network AliceNet. Alice is connected to a remote web server Bob, over a standard TCP based connection, such as HTTP, SSL etc. The attacker (Eve) has two abilities: eavesdropping and spoofing. Eve has a receiver antenna, with which she able to eavesdrop on (a small percentage of) packets sent by the access point over AliceNet. Eve in not able to inject packets into AliceNet, because of the long distance between them. Eve is also able to send raw packets into the Internet via its ISP. We assume that Eve's latency to both Alice and Bob is higher than the latency between them. Also, Eve cannot delay, drop, or otherwise affect any traffic sent in the network.

The Ack-storm attacks are based on the fact that, upon receiving a packet with the acknowledge number field (the receiver's sequence number) larger than the one sent by the receiving client, the client must, according to the TCP standard [6], resend the last sent acknowledgment packet to the other side, and discard the received packet. A design flaw in TCP causes the client and the server to be trapped in an infinite loop of sending and receiving empty acknowledgment packets.

The basic attack - Two Packets Ack Storm, as performed by the attacker, consists of three main stages:

1. Pick up (at least) one packet from a TCP connection between a client and a server.

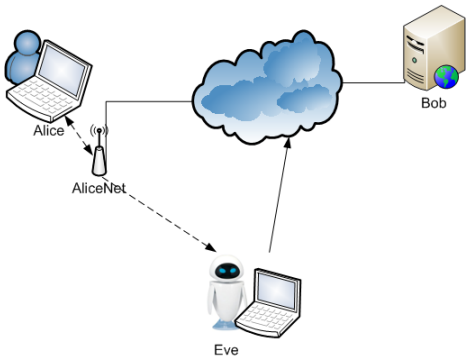

Fig. 1. Example attacker model - Alice is connected through the wireless access point AliceNet, to a remote web server Bob. Eve is able to receive occasional traffic from Alice's network. In addition, Eve's ISP does not filter traffic, so Eve is able to send spoofed packets to the Internet.

Attack	Highest Measured Ampl.	Ampl. Attacker
Two-Pkt Ack Storm	$261,000$	WitM
N Ack Storm	$261,000$	WitM
Everlasting Ack Storm	$400,000$	WitM
Opt Ack [7]	251.6	Client
Smurf [8]	≤ 1000	Spoofer
DNS Amp. [8]	73	Spoofer

Fig. 2. Comparing DoS Amplification Attacks

2. Generate two packets, each addressed to one party and with sender address of the other party (i.e. spoofed). The packets must be inside the TCP windows of both sides. The packets should have content - at least one byte of data.
3. Send the packets to the client and the server at the same time. The connection will then enter an infinite loop of sending ack packets back and forth between both parties.

The *N-packets Ack Storm* attack and the *Everlasting Ack Storm* attack offer further amplification to the Two-packets Ack Storm attack, consuming more bandwidth and increasing the duration of the session. In our experiments, we measured an amplification factor of over 400,000, when performing the Everlasting Ack Storm attack - the highest amplification rate measured until today (see Fig. 2 for comparison with existing amplification attacks).

Contributions. This paper presents the following contributions:

1. We present the WitM attack model and demonstrate how a WitM attacker can preform DoS attacks.
2. We present the Ack-storm DoS attacks. These are powerful attacks, requiring low resources (low probability to intercept packets from the network, low bandwidth requirements) from the attacker and providing the highest amplification factor measured until today.
3. The Ack-storm DoS attacks demonstrate an advantage of the use of IPSec over the use of TLS/SSL. SSL connections are vulnerable to the Storm attacks, and even help the attacker target the servers and not the web proxy. IPSec, on the other hand, is immune to the attack, as it does not reveal TCP connection details to an eavesdropper attacker.

2 Two-Packets Ack-storm Attack

In this section we present the flaw in the TCP standard that enables the Ack-storm DoS attacks, describe the Two-Packets Ack-storm attack and explain the strengths and weaknesses of this attack.

2.1 The TCP RFC Flaw

The TCP RFC [6] defines all states and actions in a TCP connection. According to the RFC, the way to handle false data is, usually, to drop the packet. This behavior is recommended as it does not allow an attacker to trigger a response from either party by injecting false packets into the stream. However, there is one exception: When a TCP connection in ESTABLISHED state, and a packet is received with an ACK field that acknowledges data not yet sent, the client must act as follows (described in page 71 of the RFC):

1. Send an ACK (the last sent).
2. Stop processing ('drop') the segment. In particular, *ignore* the payload in the segment.

Sending an acknowledge packet in response to a malformed packet is not recommended - as such packet is clearly not a result of normal traffic. This behavior is what makes the Storm attacks possible.

2.2 Attack Description

To initiate the Two-Packets Ack-storm the attacker sends two packets containing data: one to either side of a TCP connection. The attack uses the RFC flaw described above in order to cause the client and server to send false acknowledgment packets back and forth. No additional data could be sent once the attack takes place: every packet sent from now on will contain ack number higher then the one the receiving party has, and hence will only cause generation of another (malformed) ACK. If either side trys to send additional data over the channel (assuming the TCP send window is not full), the packets will increase the strength of the attack by creating additional 'sub sessions' of acknowledge packets (additional explanation can be found in the next section). Since according to the standard such packet must be dropped, and its data discarded, neither side can increase its sequence number, making it impossible for the sequence numbers to re-synchronize.

Figure 3 demonstrates the message passing between the client, server and attacker. By sending packets to both sides simultaneously, the attacker raises both the client and server's reserved ack numbers. This will make it impossible for them to overcome the false data sent, as all packets sent from that point will be considered false due to the ACK field being too high.

By sending a minimum of two packets, the Two-Packets Ack-storm attack can cause hundreds of thousands of acknowledgment packets to be sent over a

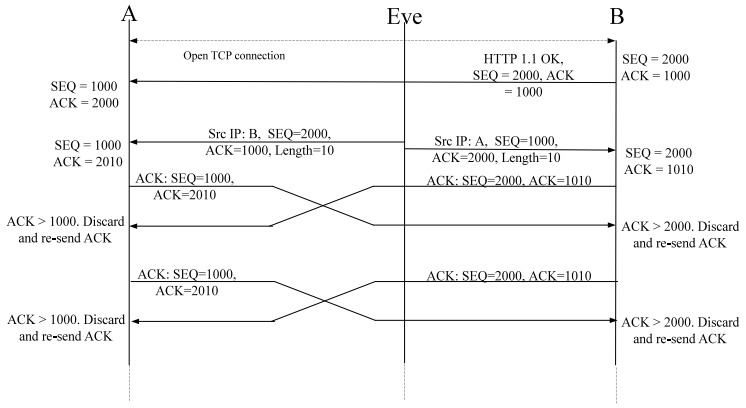

Fig. 3. The Two-Packets Ack-storm attack. Numbers are for illustration only. The attacker sends both the client (impersonating as the server) and server (as the client) a message with length 10. Both sides send an ACK, while advancing their ACK number by 10. When the packets arrive at the other side, they contain an ACK field higher than the actual data sent. The client and server then send (according to the standard) the last ack sent by them, which triggers the loop all over again.

single TCP session. The figures, presented in the experiments section (Tab. 2) show that by sending only two packets, each with the minimal length of an Ethernet packet size (64 bytes), we can cause an amplification factor of over 261,000 times the original sending size.

The attack scenario is illustrated below (and in Fig. 3); to illustrate, we assume initial values of $A.SEQ = 1000(= B.ACK)$, and $B.SEQ = 2000(= A.ACK)$.

1. Eve sends A and B packets of length 10, each on behalf of the opposite side.
2. Upon receiving the packet, A advances A.ACK to be 2010, and sends an ack to B. B advances B.ACK to be 1010 and sends an ack to A.
3. When B receives a packet with $A.ACK = 2010$, when $B.SEQ = 2000$, he acts according to the standard: discards the packet and re-sends A the ack (in which $B.ACK = 1010 > A.SEQ$). A does the same, as it received a packet from B with $B.ACK = 1010$.
4. Both A and B receive packets with the ACK number bigger than their SEQ. The behavior in step 3 is performed again.
5. The loop continues when both parties keep receiving packets with an ACK larger than their sequence numbers, stopping only when both packets are dropped, or when one side reaches a timeout and ends the connection by RST.

2.3 Analysis

The attacker sends one packet to the client and one to the server, both with length of the minimum Ethernet packet size - 64 bytes. Each of the two packets

sent by the attacker causes ACK packets (each of length 64 bytes) to be sent back and forth until the connection is terminated by the server, after $Time_R$ seconds. We are using the fact that the minimal length of an Ethernet packet is 64 bytes, while the size of an empty TCP packet is only 60 bytes. This allows the attacker to send up to 4 bytes of data without adding to the length of the packet.

The two packets sent at the beginning of the attack are sent back and forth between the client and server, creating two 'sub sessions' of traveling packets. The attack continues until a RST packet terminates the attack after the maximal number R of retransmissions is sent; let $Time_r$ denote the time of the rth retransmission (and $Time_R$ the time of the last retransmission before reset). During the total time of the attack, i.e., ($Time_R$), these two 'sub sessions' of packets would have caused a total of $128 \times \frac{R}{\rho}$ bytes sent, where ρ denotes the round trip time (RTT). Notice that the shorter ρ (the RTT between the client and server), the more effective the attack.

Since the attack interrupts an active session, altering the sequence numbers as it does so, acknowledgments of already sent packets are dropped. Therefore, the unacknowledged packets will be retransmitted by the sender. Since no retransmission will succeed, the sender will eventually give up and abort the connection.

Table 1 shows the retransmission scheme of Apache web server - the most common web server in the Internet today. Transmission times are all based on our experiments. From the table we can see that after $Time_R$ =225 seconds the server resets the connection. The server would have retransmitted ten times, in each the time waited between retransmissions is roughly doubled[1].

Table 1. Apache Server Retransmissions During the Attack

Retr. Attempt (r)	1	2	3	4	5	6	7	8	9	10	R=11
'sub sessions'	3	4	5	6	7	8	9	10	11	12	13
$Time_r(sec.)$	0	0.24	0.68	1.56	3.32	6.84	13.88	27.96	56.12	112.44	224.88
$Time_r - Time_{r-1}$ (sec)	0	0.24	0.44	0.88	1.76	3.52	7.04	14.08	28.16	56.32	112.44

Each retransmission packet that the server sends contains an acknowledgment number higher then the client's actual sequence number. Therefore, each retransmission attempt that the server sends starts a new 'sub session' of acknowledge packets sent between the client and server. Since the server makes 10 retransmission attempts, by the end of the last timeout there are 13 'sub sessions' of packets traveling back and forth. We mark $T_r = Time_r - Time_{r-1}$

[1] The retransmission policy of Microsoft IIS server is different: an IIS server will attempt a retransmission once every ten seconds, and initiate a connection abortion after sixteen unsuccessful retransmission attempts. In the analysis, we use the Apache retransmission properties, since it is more common, but the calculation could easily be modified to fit IIS (and other servers).

as the current retransmission duration, ρ being the RTT, R the maximum retransmissions before connection abortion (for Apache R=11). The amplification factor that the attacker achieves, with the first retransmission sent roughly at the beginning of the attack, is therefore:

$$Amp_{Two}(R, T, \rho) = \frac{1}{2} \times \sum_{r=1}^{R} \left(\frac{T_r}{\rho} \times (r+2) \right) \tag{1}$$

2.4 Experiments

In this section we present the results achieved both when we tested the Two-Packets Ack-storm attack in the lab, and when we tested the attack on popular sites in the Internet. We tested the attacks on both HTTP and HTTPS sites, using both Apache and IIS web servers. The results from the experiments can be seen in Table 2. In the tests we measures the RTT to the site, the number of packets the attack generated and the time passed until the server reset the connection[2].

Table 2. Comparison of 2-Packet Storm Attacks on Different Sites

Site	Total Packets	Dur. (sec) ($Time_R$)	RTT (sec) (ρ)	Server Type	Total Bytes	Ampl.	Ampl. By Analysis
live (SSL)	13,000	229	0.270	IIS	832,000	7,500	5,002
oranim	20,000	120	0.180	IIS	1,280,000	10,000	16,000
yahoo	50,000	225	0.170	Ap.	3,200,000	25,000	25,415.5
facebook	60,000	225	0.160	Ap.	3,840,000	30,000	27,004
google	140,000	225	0.110	Ap.	8,960,000	70,000	39,299
bbc (uk)	190,000	225	0.060	Ap.	12,160,000	95,000	72,000
il.msn (p)	234,000	225	0.030	Ap.	14,976,000	117,000	144,098
bing (p)	320,000	225	0.018	Ap.	20,480,000	160,000	240,163.5
Lab	522,000	225	0.001	Ap.	33,408,000	261,000	4,322,944

In Tab. 2 we can see that while attacking sites running Apache, the time until termination of the attack remains constant - 225 seconds. This value is the connection abortion after maximum failed retransmissions. Once the attack takes place, no data is acknowledged in the session. That causes the server (usually the one sending the data over HTTP sessions) to retransmit the data 10 times, when each time the retransmission timeout doubles. Table 1 shows us the times of the retransmissions in relation to the beginning of the attack, and the termination of the connection occurred after the 10th timeout expired.

[2] We focused on Apache servers, because they are the most common. IIS and SSL(live.com) were tested for the attack but comparative research was not done on them.

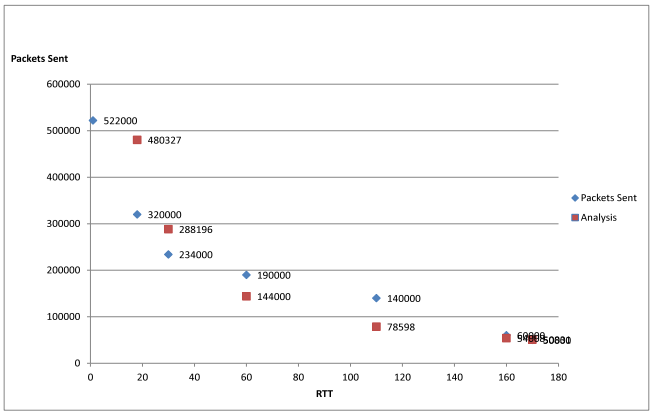

Fig. 4. Attack results when attacking various Apache-based sites ($Time_R = 225$), in comparison to analysis. We measured the number of packets sent and compared to the analysis. The graph shows the correlation between the analysis results, and the ones achieved in real attacks. Differences can occur due to packet loss/duplication, retransmission time deviations etc.

The Two Packet Storm attack presents a substantial amplification factor to the data sent by the attacker. The attack, however, has two main limitations:

1. The attack causes a constant number of 'sub sessions' , consuming a limited amount of network resources. If an attacker wishes to attack a high bandwidth target, he would have to use a large number of connections.
2. This attack is time limited, since a server will terminate the connection after reaching the maximum failed retransmissions attempts. After $Time_R$ seconds, the server will abort the connection, terminating the attack in the process. In order to attack a target for a time larger than $Time_R$, he would have to start new attacks to replace the old ones.

In the following sections we present two variations of the Two-packets Ack-storm DoS attack, which address the limitations described above.

3 The N-packet Ack-storm DoS Attack

The N-packet Ack-storm DoS attack enables the attacker to increase the number of packets sent over an attacked session. When using the N-packets Ack-storm DoS attack, the attacker can consume all of the bandwidth available for the session.

By Injecting additional ack packets to an attack already in progress, the attacker increases the bandwidth consumed by the attack. For every additional acknowledgment packet the attacker injects into the stream, another 'sub session' is created, and the client and server start passing the packet between them. This method bypasses the latency limitations of the client and server, since

more 'sub sessions' simulate a shorter distance between the client and server: for the affective RTT to decrease by 0.5, the attacker doubles the number of 'sub sessions' in the connection.

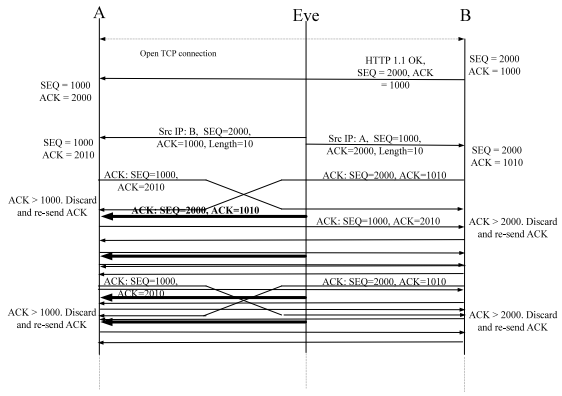

Fig. 5. The N-packets Ack-storm DoS attack. The storm packets sent by the attacker are in bold. Every new packet triggers another 'sub session' of acknowledge packets sent back and forth.

3.1 Analysis

In order to maximize the effectiveness of the additional 'sub session' he creates, the attacker sends the additional Storm packets at the beginning of the attack. The number of 'sub sessions' at the end of the attack, for 2 packets sent in order to trigger the attack, N additional Storm packets and retransmission generated packets is $2 + N + R$. The amplification achieved would be, for R maximum retransmissions and 64 bytes acknowledgment packet length:

$$Amp_N\left(N, R, \rho\right) = \frac{1}{2+N} \times \sum_{r=1}^{R} \left(\frac{T_r}{\rho} \times (r + 2 + N) \right)$$

The attack minimizes the sessions needed to reach a high bandwidth consumption. Using storm packets, we were able to trigger *485,000* packets sent over a single session (with Yahoo.com), when the Two Packet Storm generated only 50,000.

When reaching the bandwidth limitations of a connection, packet losses will start to occur. Each packet loss will end the 'sub session' of that packet in the stream, while the rest of the 'sub sessions' will continue executing.

Table 3 presents the amplification results when attacking Yahoo.com. Number of packets increases as a function of packets sent, until reaches a point where packets losses start occurring (at about 42 Storm packets sent). Notice that

packet losses do not stop the attack, but only limit the maximum bandwidth it can consume. We did not send over 42 packets in order to avoid causing real harm to users in the network, as would result from causing congestion on a live network).

Table 3. The N Packet Storm on Yahoo.com

Storm Packets Sent	Packets In Session	Attack Duration (T)	Bandwidth Consumption (Max.) $\left(\frac{Bytes}{Sec}\right)$	Amplification Factor
2 (Two Packet Storm)	50,000	225	4,517	25,000
6	103,000	225	6,023	17,166
10	153,000	225	7,579	15,300
22	283,000	225	12,047	12,863
42	485,000	225	19,576	11,547

While bandwidth consumption increases as a function of the Storm packets sent, the amplification factor decreases. The attacker has to send an additional packet for every 'sub session' he wants to create, but the number of retransmissions (and the number of 'sub sessions' they create) remains the same. The disadvantage of lower amplification factor is balanced by the lack of need to manage multiple attacks in order to achieve the same amount of bandwidth usage.

4 Everlasting Ack Storm Attack

The main limitation of the two attacks presented so far is the the maximal connection duration $Time_R$. The attacker can avoid a connection abortion by artificially sending data over the channel, preventing the client or server from reaching a timeout. The Everlasting Ack Storm data packets should contain at least one byte of data, which will not add to their size because it is below the Ethernet packet size minimum. The packets should be sent at least every $Time_R$ seconds, in order to avoid the timeout. When data is being sent over the connection (however considered retransmission by both the client and server), the connection timeout is not triggered, allowing us to continue the attack until reset by application layer or any other network entity.

4.1 Analysis

The number of initial packets sent by the attacker is the same as the Two Ack Storm attack. The additional packets are sent every $Time_R$ seconds in order to maintain the connection. The amount of total data sent over the channel is composed of the initial $Time_R$ seconds, in which the amount of packets is

identical to those of the Two Packet Storm attack, and the following minutes, in which the attack continues with the number of 'sub sessions' increases for every storm packet sent. The amplification achieved when sending Eve_R storm packets (for R maximum retransmissions of server):

$$Amp_{\infty}\left(R,T,\rho\right)=\frac{1}{Eve_R}\left(\times\sum_{r=1}^{R}\left(\frac{T_r}{\rho}\times r+2\right)+\sum_{r=R}^{Eve_R}\left(\frac{Time_R}{\rho}\times\left(r+2\right)\right)\right)$$

In Tab. 4 we show the different abilities the attacker can achieve by using the attacks described above. Notice that the amplification factor when sending storm packets without data decreases, but the total bandwidth increases as a function of the packets sent. Also notice that when sending storm packets with data, the attacker increases both the duration of the attack (by avoiding timeout), and the bandwidth consumed by it (as it opens another 'sub session')).

Table 4. Amplification Types - Assuming No Bandwidth Limitation

Attack type	Data Sent	Attack Intervals	Attack Duration	Time Until Full Bandwidth Consumption	Ampl. Factor
Two	128 B	Once	$Time_R$	Never	Amp_{Two}
N	$128\times Q$	Once	$Time_R$	Immediate	$Amp_N \leq Amp_{Two}$
Everl.	$128\times L$	Every T_{rst}	$Time_R\times L$	$(Time_R)\times\left(\frac{BW}{64}-(2+R)\right)$	$Amp_{\infty} >> Amp_{Two}$

Using Storm packets with data, sent every 1 minute, we maintained an attack on Google.com for over 26 minutes, causing over 10,000,000 packets (over 640,000,000 bytes of data) to be sent before terminating the connection - an amplification factor of 400,000. We terminated the attack as part of our policy to avoid causing damage to the attacked sites.

5 Preserving the Attack During Losses

When the attack reaches the bandwidth limitations of the channel, packet losses start to occur. For every dropped ack packet, one 'sub session' is stopped and the attacks consumed bandwidth drops by $\frac{64}{\rho}$Bps. In order to maintain the bandwidth of the attack, the attacker must generate an storm packet for every dropped ack packet.

When a TCP connection encounters losses, the TCP window size decreases rapidly. In the full version of the article, we present an experiment demonstrating the depredation in TCP bandwidth when performing an attack.

6 Conclusion and Future Work

In the article we presented the the WitM attacker model. We showed the Storm attacks, which can cause DoS to network infrastructure, as well as individual

web sites and services. We showed both in analysis and in experiment results the high amplification factors the Storm attacks achieve.

The Storm attacks are just the beginning of a wide rage of attacks possible for the WitM attacker. Zombie-based DoS attacks can be accomplished by a WitM without the need to control the node itself, but only by using its connection details. Injection attacks are also possible to the WitM attacker, however the ability to inject the data in the correct timing requires further work, due to the high latency the attacker has. Finding additional DoS attacks and uses for a WitM attacker will add additional impact for the WitM attacker model.

Acknowledgments Many thanks to Charlie Kaufman, Amit Klien and Ben Laurie for their important feedback and encouragement. Amit also introduced us to the earlier work discussing Ack storms (as an undesirable side-effect of TCP hijacking attacks), e.g.[4]

References

1. Borella, M., Grabelsky, D., Lo, J., Taniguchi, K.: Realm Specific IP: Protocol Specification. RFC 3103 (Experimental) (Oct 2001), http://www.ietf.org/rfc/rfc3103.txt
2. Chandra, P.: How To Make A WiFi Antenna Out of A Pringles Can. makeuseof.com (August 2009), http://www.makeuseof.com/tag/how-to-make-a-wifi-antenna-out-of-a-pringles-can-nb/
3. Herzberg, A., Shulman, H.: Stealth DoS attacks on secure channels. In: NDSS (March 2010)
4. Joncheray, L.: A simple active attack against TCP. In: Proceedings of the 5th Symposium on UNIX Security. pp. 7–20. USENIX Association, Berkeley, CA, USA (Jun 1995)
5. Lam, Antonatos, Akritidis, Anagnostakis: Puppetnets: Misusing web browsers as a distributed attack infrastructure. In: SIGSAC: 13th ACM Conference on Computer and Communications Security. ACM SIGSAC (2006)
6. Postel, J.: Transmission Control Protocol. RFC 793 (Standard) (Sep 1981), http://www.ietf.org/rfc/rfc793.txt, updated by RFCs 1122, 3168
7. Sherwood, Bhattacharjee, Braud: Misbehaving TCP receivers can cause internet-wide congestion collapse. In: SIGSAC: 12th ACM Conference on Computer and Communications Security. ACM SIGSAC (2005)
8. Vaughn, R., Evron, G.: DNS amplification attacks. ISOTF (Mar 2006), http://www.isotf.org/news/DNS-Amplification-Attacks.pdf
9. Wong, M., Clement, A.: Sharing wireless internet in urban neighbourhoods. In: Steinfield, C., Pentland, B.T., Ackerman, M., Contractor, N. (eds.) Communities and Technologies 2007, pp. 275–294. Springer London (2007), http://dx.doi.org/10.1007/978-1-84628-905-7_15, 10.1007/978-1-84628-905-7_15
10. Wu, B., Chen, J., Wu, J., Cardei, M.: A survey of attacks and countermeasures in mobile ad hoc networks. In: Xiao, Y., Shen, X.S., Du, D.Z. (eds.) Wireless Network Security, pp. 103–135. Signals and Communication Technology, Springer US (2007), http://dx.doi.org/10.1007/978-0-387-33112-6_5, 10.1007/978-0-387-33112-6_5