The Coremelt Attack*

Ahren Studer and Adrian Perrig

Carnegie Mellon University
{astuder, perrig}@cmu.edu

Abstract. Current Denial-of-Service (DoS) attacks are directed towards a specific victim. The research community has devised several countermeasures that protect victim hosts against undesired traffic.
We present Coremelt, a new attack mechanism, where attackers only send traffic between each other, and not towards a victim host. As a result, none of the attack traffic is unwanted. The Coremelt attack is powerful because among N attackers there are $O(N^2)$ connections, which can cause significant congestion in the network core. We demonstrate the attack based on simulations within a real Internet topology using realistic attacker distributions and show that attackers can induce a significant amount of congestion.

1 Introduction

Over the past two decades, the Internet has become of critical importance for social, business, and government activities. Corporations depend on Internet availability to facilitate sales and the transfer of data to make timely decisions. SCADA networks often use the Internet to enable coordination between physical systems. Unfortunately, malicious parties have been able to flood end hosts with traffic to interrupt communication. In these Denial-of-Service (DoS) attacks, the network link to the server is congested with illegitimate traffic so that legitimate traffic experiences high loss, preventing communication altogether. Such a loss of connectivity can wreak havoc and translate to monetary losses[1] and physical damages. Loss of connectivity between SCADA systems can cause damage to critical infrastructures. For example, electrical systems with out-of-date demand information can overload generators or power lines. Unfortunately, a failure in a critical system may set off a chain reaction, as we witnessed during the August 2003 Northeast US blackout.[2]

* This research was supported in part by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and MURI W 911 NF 0710287 from the Army Research Office, and grant CNS-0831440 from the National Science Foundation. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, NSF, or the U.S. Government or any of its agencies.

[1] In a recent attack, a week-long botnet cyber-attack costs a Japanese company 300 million yen, see article at http://www.yomiuri.co.jp/dy/national/20080601TDY01305.htm and http://blog.wired.com/sterling/2008/06/looks-like-a-ya.html.

[2] More information on the August 2003 Northeast US blackout is available at: http://en.wikipedia.org/wiki/2003_North_America_blackout.

A commonality of past Denial-of-Service (DoS) attacks is that adversaries directly attacked the victim. Consequently, defenses that were designed to prevent such attacks aim to identify the source of excessive traffic or prioritize legitimate traffic. Since machines can insert fake source addresses, different tracing schemes have been developed to identify the origin network of malicious traffic in the hope that an ISP will "pull the plug" on malicious activities once the sources are identified. However, attackers often rely on Distributed Denial of Service (DDoS) attacks where numerous subverted machines (also called a botnet) are used to generate traffic. With a large botnet, each malicious source can generate a small amount of traffic to make it more difficult for victims to distinguish legitimate traffic from malicious traffic. To address such stealthy attacks, capability-based systems allow end hosts to identify long-running legitimate traffic, which routers prioritize for delivery. During times of heavy load, routers forward packets with the proper capabilities while dropping packets without capabilities.

Once tracing and traffic capabilities are deployed, attackers will look for new ways to launch DoS attacks. Rather than targeting endpoints or the network link heading to a victim, the attacker may aim to disrupt core network links in the Internet. Prior work has shown that disabling important links can cause substantial damage in terms of isolating parts of the Internet [1]. With enough subverted machines under control, a malicious party can generate enough traffic to choke even the largest links. For example, an OC-768 link (the largest type of link currently deployed) has almost 40 Gb/s of bandwidth. A botnet with 350,000 DSL customers spewing 128 kb/s can generate ample data (over 43 Gb/s) and overload such a link.[3] Of course, the attacker cannot just spew packets at the different ends of a crucial link. Network administrators can easily filter such attack traffic since benign customer packets rarely have routers as their destination.

Since packets directed at routers will be dropped, the attacker's next option may be to send packets towards addresses a few hops past the router. However, capability-based DoS prevention systems will thwart such an attack. The destinations will not grant capabilities to malicious sources. Routers will allow legitimate traffic to traverse the congested link and drop attack traffic that lacks the capabilities.

In this work, we investigate the efficacy of a new type of DoS attack that can elude prior DoS defenses and shut down core links (i.e., a Coremelt). To circumvent current DoS defense systems that attempt to eliminate unwanted traffic, the botnet in the Coremelt attack sends only wanted or "legitimate" traffic: connections between pairs of bots. Since in a network with N bots there are $O(N^2)$ connections among all pairs of bots, these "legitimate" flows can exhaust the network bandwidth of core network links. As a result, flows from

[3] In a more pessimistic scenario, a botnet of one million nodes with connection speeds of 1 Mb/s per node can congest 25 OC-768 links. What is even more troubling is that home network connection speeds are likely to increase further, for example in Japan and Korea 100 Mb/s connections are commonly available to home users.

legitimate clients that need to cross these congested core network links will be severely affected.

The goal of this work is to define and analyze such Coremelt attacks. We simulate such an attack with real Internet topology and routing data, and distributions of real subverted machines. This data allows us to examine how Coremelt attacks from real bot distributions would impact the current Internet.

The main contribution of this work is to present the Coremelt attack, a serious attack that is possible even in a network that only permits "legitimate" traffic, i.e., traffic that is desired by the receiver. This attack suggests that more powerful countermeasures are needed to eradicate DoS attacks.

2 The Coremelt Attack

In this section, we discuss the details of a Coremelt attack and the challenges an attacker faces when launching such an attack.

In a Coremelt attack, the attacker uses a collection of subverted machines sending data to each other to flood and disable a network link. With subverted machines sending data to each other, an attacker can elude capability- and filtering-based DoS defenses because all traffic is desired by the receiver. When the subverted machines are dispersed across multiple networks, the attacker has a greater chance of shutting down a backbone link, without crippling smaller tributary links. There are 3 steps to launching a Coremelt attack:

1. Select a link in the network as the *target link*.
2. Identify what pairs of subverted machines can generate traffic that traverse the target link.
3. Send traffic between the pairs identified in step 2 to overload the target link.

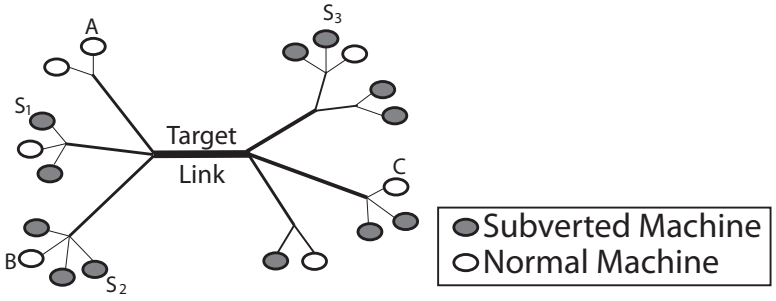


Fig. 1. Example Network Where Coremelt Would Succeed. (Note: Line thickness indicates available bandwidth.)

Figure 1 depicts an ideal setting for a Coremelt attack. The attacker will select source-destination pairs such that traffic will traverse the target link. For

example, S_1 and S_3 will send traffic back and forth, but S_1 and S_2 will not communicate. If the sum of incoming links' bandwidths is greater than the target link's bandwidth, attack traffic can flood the target link without interrupting traffic on the smaller links. When the attack is successful, legitimate nodes A and B in Figure 1 can communicate, but neither can reach C due to the congestion on the target link.

When an attacker wants to use Coremelt to disrupt a more realistic network, an attacker needs several items to prepare for the attack: knowledge of the network topology, a large botnet, and a way to generate traffic that intermediary nodes will forward. Generating a good model of the physical layer of the Internet is an open research problem. However, a botnet owner can use `traceroute` to map the paths between every pair of bots under her control. With knowledge of all $\frac{N(N-1)}{2}$ paths, the nodes simply have to decide which paths traverse the target link and only send attack traffic across those paths. Clogging a high-bandwidth backbone link requires an attacker to find significant resources. Unfortunately, real botnets on the order of 1 million nodes exist[4] and botmasters (the individuals who control a botnet) are starting to rent out botnets for hire.[5] With sufficient funds, a malicious party can rent a large enough botnet—or several botnets. Next, an attacker needs a way to generate traffic that appears normal enough that traffic filtering by the ISPs will allow it to pass. TCP is designed to reduce bandwidth usage in response to packet loss, so that traffic will simply slow down once the target link is under stress. One solution is to use non-conforming/greedy traffic that is labeled as TCP, but fails to behave according to congestion-avoidance [2]. UDP traffic is another option, assuming ISPs do not throttle that traffic.

The remainder of this work is dedicated to simulation of Coremelt to evaluate its threat and discussion of potential solutions. Before presenting simulation results, we describe the simulator and the attacker and network models we use.

3 Simulation Setup

The goal of this work is to evaluate the strength of a Coremelt attack under realistic conditions. Can a botnet generate sufficient traffic to congest a backbone link? Will the attack also congest smaller links, or will only the performance on the target link degrade? How large of a botnet is needed to launch such an attack?

Given the legal and ethical issues surrounding DoS attacks, rather than renting a botnet and attacking the Internet, we simulate the attack using realistic network topologies and attackers. In this section, we describe the data we use to model the network topology and attacker. We also describe the simulator we use

[4] Some professionals claim the Storm worm botnet reached 1 to 50 million nodes at one time. `http://www.informationweek.com/news/internet/showArticle.jhtml?articleID=201804528`

[5] `http://www.usatoday.com/tech/news/computersecurity/2004-07-07-zombie-pimps_x.htm`

to model the flow of traffic in our simulated Internet. We conclude this section with the different metrics we use to quantify the success of a Coremelt attack.

3.1 Network Model

We use Autonomous System (AS) level information to build a graph and select routes between nodes that match the topology of the Internet and likely routes Internet traffic would take. We use the CAIDA AS relationships Dataset [3] from January of 2009 so our model can take into account both the presence of links and the priority of different links when routing traffic in the Internet. Our network model also uses AS information to dictate the resources available for a given AS to handle traffic.

For our network model, we build a graph based on the ASes in the Internet. Each node is an AS and an edge between two nodes represents an AS relationship. AS relationships can be one of four types: provider, customer, peer, or sibling. A provider is a larger AS that allows smaller customer ASes to reach a larger fraction of the Internet. Customers pay providers for these services based on the amount of bandwidth used. To reduce fees, ASes often peer with other ASes and exchange traffic for free to increase connectivity (i.e., a back-up link if a provider fails) or to reduce costs (i.e., when peered, traffic between customers of ASes A and B can go directly to each other rather than through a common provider AS C). Sibling ASes are two ASes owned by the same company. Assuming each AS is a single node in the network graph is a significant simplification given some ASes have large internal networks. However, this simplification provides an approximation of the structure of the Internet and more relevant results than those found using toy networks or generated graphs.

To determine the path that traffic will take between two nodes in the graph, we find the shortest route (in terms of number of AS hops) that does not violate routing policy. This requires that peering ASes will only accept traffic that is destined to their customers. For example, consider the scenario where ASes A and B are peers and have different providers such that B's provider has a shorter route to a destination D. When AS A wants to send traffic to D, A will send traffic to its provider, rather than routing the traffic through B to achieve the shorter path in terms of hops. Once the shortest AS policy abiding path is found, we consider it fixed for the remainder of our simulation. For future work, we plan to investigate how changing routes based on congestion can redistribute traffic and help prevent Coremelt attacks, or if changing routes will simply redirect attack traffic to a new bottleneck link which will subsequently fail.

Different links in the Internet have different capacities. However, there is little information available about the bandwidth of a backbone link within an AS. When simulating the Coremelt attack, we want an accurate estimate of how much traffic an AS can support at a time. For example, we want to know the capacity of AT&T's optic cable between the US and Europe. Obviously, that bandwidth is different from the bandwidth available on the major link of a regional ISP. AS degree is a pragmatic way to estimate the capacity of an AS. An AS's degree is the number of other ASes that directly communicate with the

given AS. The more clients an AS supports and the more peers an AS shares traffic with, the more traffic that AS can support. In our simulations, we consider a number of different capacity functions.

- **Uniform:** every AS can support the same amount of traffic. This scenario is inaccurate, but represents a worst-case scenario for high degree ASes under a Coremelt attack. With multiple incoming links of the same bandwidth, high degree ASes are more likely to fail under Coremelt.
- **Linear:** the bandwidth an AS can support grows linearly with AS degree. This is a best-case scenario for high-degree ASes. When an attacker aims to disrupt a major AS, incoming ASes with smaller degrees will be congested and drop traffic such that the high-degree AS can support the aggregate of the incoming traffic. This model is unrealistic due to the cost of increasing bandwidth. Additional interfaces on a router allow an AS to contact a different AS and increase its degree, but increasing the bandwidth within the AS requires purchasing additional links and/or upgrading existing links.
- **Step:** the most realistic of our settings assumes that ASes fall into different classes of resources based on their degree. We analyze the sensitivity of the results under the step model using two different step functions.

In Section 3.5, we describe the actual values we use in each scenario.

3.2 Attacker Model

In a Coremelt attack, an attacker is limited by three key properties: the size of the botnet, the distribution of bots, and the amount of traffic each bot can generate.

In our simulations, we test a range of botnet sizes and traffic generation capabilities (see Section 3.5 for specific numbers) to test Coremelt's sensitivity under varying conditions. However, it is difficult to determine a realistic distribution of bots. Coremelt has the greatest chance of success when bots are evenly distributed across the Internet. However, instead of assuming some distribution of bots over the Internet, we use records from real attacks. Once we know the distribution of subverted machines, we can scale the botnet to various sizes. For example, if 50 bots from a 1,000 bot botnet reside in AS M, we simulate a botnet of size 1,000,000 by assigning 50,000 bots to AS M. Once we have a bot distribution and have scaled the botnet to a given size we vary the traffic generation capability of the bots to evaluate when Coremelt will succeed to congest a link.

In our simulations, we examine two sets of subverted machines: machines infected with CodeRed and a set of machines used to launch a DDoS attack against a computer at the Georgia Institute of Technology. For the remainder of this paper, we refer to the data sets as CodeRed and GT-DDoS, respectively. The CodeRed set comes from CAIDA data that lists the IP addresses of machines infected with CodeRed scanning for other vulnerable machines in July of 2001 [4]. There are 278,286 infected machines that we can associate with 4746 ASes in our network model. CodeRed was a worm that infected machines running Microsoft's

IIS web server. However, the data still provides a rough approximation of the distribution of vulnerable hosts on the Internet. If admins in a network fail to patch servers, the admins have likely neglected to patch clients in that network. One disadvantage to this data set is that it fails to represent the networks without servers. Such networks may contain a large number of vulnerable clients, but no servers. Our second data set contains real botnet data and thus can provide a realistic distribution of vulnerable machines. This set includes 5994 unique IPs that we can associate with 720 ASes in our network model. Even though this is a relatively small botnet, we scale this number while maintaining the distribution of bots to simulate larger botnets.

3.3 Simulation Methodology

In this section, we explain how we integrate our network and attacker models and how we simulate the flow of traffic through the network in a discrete fashion.

In our simulation, each node in the network is an AS. Each AS has 0 or more bots and can support different amounts of traffic, depending on the function used to simulate AS resources (i.e., uniform, linear, or step). Based on the CodeRed or GT-DDoS data, we know each AS contains some number of bots (B). We scale the botnet by a factor, F, so the number of bots in a given AS changes from B to $\lfloor FB \rfloor$. This assures the same distribution of bots across simulations, while changing the effective size of the botnet. For our simulations, each bot can generate a fixed amount of traffic T. Rather than increasing the memory usage as T increases, we normalize the resources of the ASes with respect to T so each bot only generates one piece of data per time interval. For example, if we assume one bot can generate 14 kilobits per second and an AS can handle 1 gigabit per second, the AS is scaled to handle 74898 meta packets per interval ($74898 = \lfloor \frac{1 \cdot 2^{30}}{14 \cdot 2^{10}} \rfloor$).

Our simulator works in two steps: initialization and traffic routing. Initialization handles defining routes and AS statistics. Defining routes involves finding the different routes in the network and selecting which routes an attacker will use to attack a given target. The simulator then assigns the number of bot sources to each AS based on the original botnet distribution and the input scale factor. Finally, the simulator allocates buffers for each AS to store packets where the size of the buffer is based on how many packets the AS can handle in one second.

Our simulator is a discrete time simulator where during interval i the ASes forward packets they received in interval $i-1$ and collect packets to forward during interval $i+1$. At the start of an interval, an AS generates $\lfloor FB \rfloor$ (the total number of bots in that AS) packets, randomly selects ASes with bots as the destinations for the packets such that the packets will traverse the target AS, and stores the packets in an incoming buffer. This generation in interval i and sending in interval $i+1$ simulates the machines in the AS generating the packet, rather than the routers in the AS. If bots in an AS generate more packets than the AS can support, the AS drops the extra packets. Next, the AS forwards the packets received during interval $i-1$ to the next hop in each packet's path. When an AS receives a packet from another AS, the packet is placed in the incoming buffer to

be either forwarded to the next hop in the path or delivered—if the destination is in this AS—in interval $i+1$. If the AS's incoming buffer is already full when it receives a packet, the AS randomly selects a packet from the buffer, drops that packet, puts the newly received packet in the buffer, and notes the overload for that AS. In our simulator, there is no legitimate traffic that flows between nodes; all of the traffic flows between bots. The introduction of legitimate traffic could hinder or help a Coremelt attack. Additional traffic could cause congestion on tributary links and prevent attack traffic from reaching the target link, reducing the impact of a Coremelt attack. However, the majority of legitimate traffic will likely use congestion avoidance, allowing greedy/non-conforming attack traffic—which never backs off—to proceed unhampered to the target link. The addition of legitimate traffic on the target link will increase the chance of a successful Coremelt attack since additional traffic on the target link increases the chance of the link exceeding its limit.

For each scenario, we simulate the generation and forwarding of packets for 50 intervals. We tested longer simulations, but given the limited diameter of the network, packets either overload the target within a short period of time or the attack fails.

3.4 Metrics

The goal of this work is to measure the success of a Coremelt attack under varying conditions. To quantify the success of an attack, we use two metrics: destructiveness and stealthiness.

Destructiveness indicates if a Coremelt attack is able to overload different target ASes in our simulation. Since Coremelt aims to attack the core of the Internet, we define destructiveness as the fraction of the top ten ASes an attacker can congest one at a time with a given botnet size and traffic generation capabilities. A destructiveness of 0.3 means an attacker can shut down 3 of the top 10 ASes.

Stealthiness indicates how many non-target ASes are impacted by a Coremelt attack. The goal of Coremelt is to shut down the target while minimizing impact on the rest of the network. Additional congested ASes increase the chances of ASes reacting to the congesting flows (e.g., dropping packets) or tracing the attack traffic back to the bots. To measure the stealthiness of Coremelt, we record the sum of non-target or collateral ASes that are also congested when individually attacking the top 10 ASes. For example, if the attacker happens to congest 3 additional ASes while attacking each of the top ten ASes, the number of congested collateral ASes is 30. We count a top ten AS as part of the congested collateral ASes if it is not the current target.

An attacker's goal is to achieve a high destructiveness while maintaining stealthiness by limiting the number of ASes that experience collateral damage.

3.5 Simulation Parameters

We now present the different values we use during simulation for traffic generation, botnet size, and AS resources.

We take a conservative approach to bots' traffic generation abilities and test botnets where all nodes are connected via dial-up modem or DSL. Specifically, we assume bots can generate either 14 kilobits per second or 128 kilobits per second. Given the proliferation of high speed links available for home users, these are conservative values.

During our simulations, we test a range of botnet sizes. We sweep through a range of values to determine the smallest botnet that can shut down the top ten ASes intentionally and the smallest botnet such that there are zero ASes experiencing collateral damage.

During simulation we varied the ASes' resources based on the three bandwidth models in Section 3.1: uniform, linear, and step. In the uniform model, we assume every AS backbone has a fixed bandwidth. We run two sets of simulations to determine the sensitivity to resources selected. The first set assumes each AS backbone can handle 2.5 Gb/s while the second assumes 5 Gb/s. Under our linear model, an AS with degree d has d OC-12 links for a total bandwidth of $d \cdot 601$ Mb/s.[6] For example, an AS of degree 3 can support 1,803 Mb/s ($3 \cdot 601$ Mb/s). Our last model uses a step function to determine the bandwidth of an AS based on its degree. Table 1 contains the list of different classes of ASes in our step function and the number of ASes in each class. To test the sensitivity of the attack under the step model to our function, we also run additional simulations where ASes with degrees of 1000 or more have twice the resources. Note, by giving the target ASes (the top ten ASes) significantly more bandwidth than the rest of the ASes, we are reducing the chance of Coremelt destructiveness and increasing the chance of collateral ASes suffering congestion.

Degree (d)	Link	Bandwidth	# of ASes
$d = 1$	OC-12	601.344 Mb/s	11,042
$1 < d < 10$	OC-48	2,405.376 Mb/s	18,083
$10 \leq d < 999$	OC-192	9,621.504 Mb/s	1475
$d \geq 1000$	OC-768	39,813.12 Mb/s	10

Table 1. The step function we use to define resources for ASes based on degree.

These network resources may be less than what ASes can support in real life. However, we have also underestimated the traffic generation abilities of bots. Smaller values for both of these parameters cancel each other to provide a realistic simulation (i.e., attackers generating more traffic that traverses a network with more resources will experience similar results).

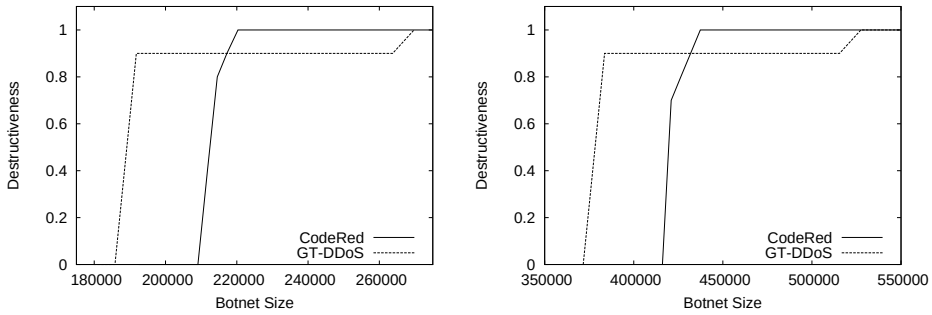
[6] For the exact bandwidth for the different levels of optical carrier links see `http://en.wikipedia.org/wiki/Optical_Carrier`.

4 Simulation Results

Our simulation results indicate that networks where resources follow the uniform and step models are vulnerable to the Coremelt attack. The major difference is the ability to focus an attack. In uniform networks, an attacker can precisely attack a single core AS. However, in networks that follow the step model, an attacker will congest additional ASes when targeting some core ASes. If resources are more like the linear model, an attacker with a very large botnet can launch a successful Coremelt attack, but shuts down the majority of the network in the process causing substantial collateral damage.

4.1 Uniform Network

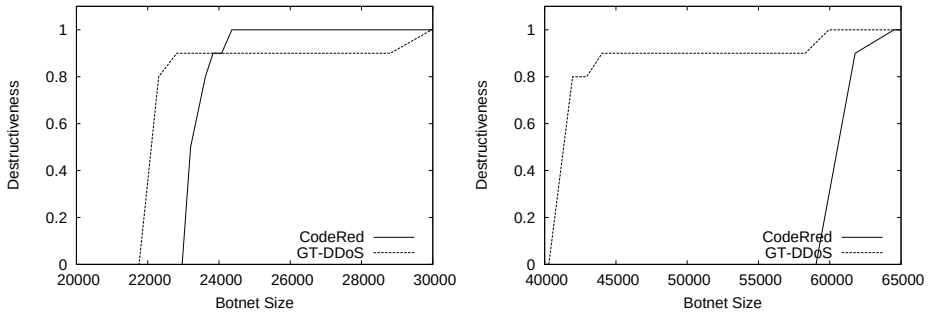
The destructiveness of a Coremelt attack in an uniform network is shown in Figures 2 and 3. For the uniform network model, the Coremelt attack is a serious threat. With botnets in the shown ranges, a Coremelt attack is very stealthy and the number of congested collateral ASes is 0. When the resources of the target ASes double, a successful attack requires roughly twice as many bots.



(a) 2.5 Gb/s (b) 5.0 Gb/s

Fig. 2. Results when simulating an attacker with 14 kbps per bot when ASes have uniform resources.

One unexpected result is that destructiveness is not a binary result. One may expect that as soon as an attacker can generate enough traffic to attack one of the top ten ASes, all of the other top ten ASes should be vulnerable. The reason for this lies in the distribution of the bots across different ASes. With a nonuniform distribution of bots, certain targets face more traffic when facing the same size botnet. For example, with X total bots, some fraction of the bots, f_i, can send packets to each other such that traffic traverses the target AS i. With a different target AS j, some different fraction f_j is able to send packets to each other which traverse the target. If $f_i > f_j$, a smaller botnet can successfully attack AS i but fail when targeting AS j.



(a) 2.5 Gb/s (b) 5.0 Gb/s

Fig. 3. Results when simulating an attacker with 128 kbps per bot when ASes have uniform resources.

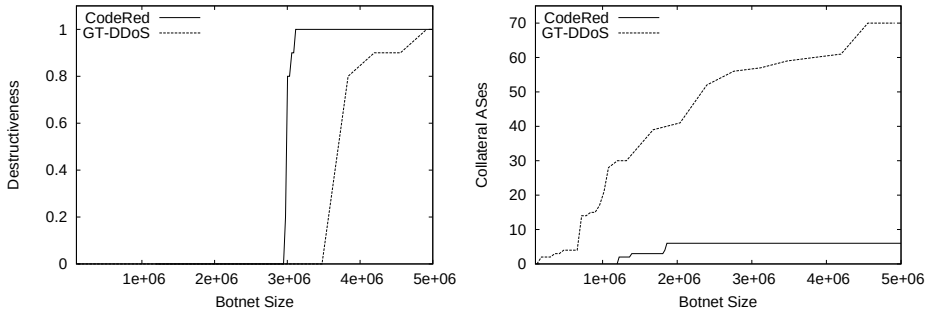
When the resources of the ASes double, the size of the botnet needed to launch a Coremelt also doubles. However, the way we scale a botnet produces some unexpected results for the attacker with greater traffic generation capabilities. Looking at the 14 kbps attacker (Figure 2), an attacker under both CodeRed and GT-DDoS distributions needs roughly twice as many bots when resources change from 2.5 Gb/s to 5.0 Gb/s for each AS. With 128 kbps traffic generation capabilities and the CodeRed distribution, an attacker needs 2.5 times the bots to achieve the same level of destructiveness when AS resources are doubled. The flooring function used to scale the botnet from 278,000 bots down to tens of thousands causes this anomaly. When the botnet scaling factor is small, the number of bots in an AS (and the number of ASes with bots) changes in set increments—rather than a linear fashion—as the overall size of the botnet changes. As such, the number of bots that can send packets across the target ASes doubles while the total number of bots changes by a factor 2.5. The GT-DDoS data set originally has roughly six thousand total bots so scaling from 6,000 to 50,000 provides a relatively smooth growth. As such, attacks on a 5 Gb/s AS take twice as many bots as attacks on a 2.5 Gb/s AS.

4.2 Linear Network

With a linear model for network resources, the Coremelt attack fails under reasonable scenarios. The top ten ASes have such a large degree that their resources can handle incoming traffic for any reasonable size botnets with our traffic generation capabilities. When an attacker tries to launch a Coremelt attack in such a network, a large number of collateral ASes will become congested. With the linear model and a non-uniform distribution of bots, an attacker may flood every AS on the path to the target AS and still fail to shut down the target.

4.3 Step Network

In the more realistic step network model, the Coremelt attack can successfully target core ASes. However, the distribution of the bots plays an important role when considering congested collateral ASes. Greater attack traffic generation capabilities allow an attacker to succeed with fewer bots, but congest the same number of collateral ASes. With bots spread through more ASes, an attacker requires fewer bots to successfully launch an attack or can use the same number of bots and congest fewer collateral ASes. Figures 4 and 5 show the destructiveness and the number of congested collateral ASes under the step model for 14 kbps and 128 kbps traffic generation capabilities, respectively. The results from simulation of the step model where we double the resources for ASes with degrees of 1000 or more are shown in Figure 6. These latter results provide strong evidence that having a botnet spread over more ASes is an advantage when launching a Coremelt attack.

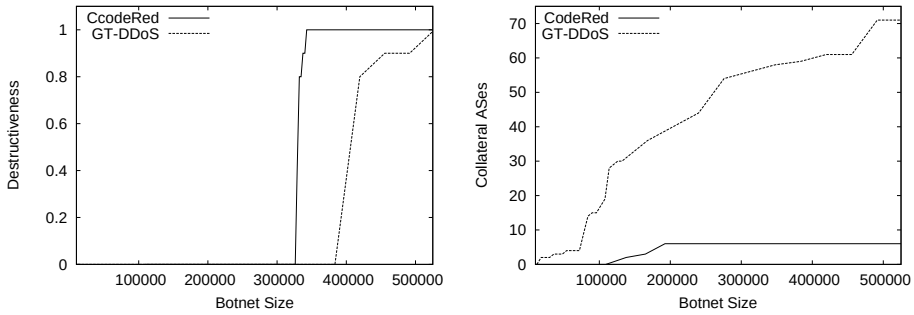


(a) Destructiveness (b) Congested Collateral ASes

Fig. 4. Results when simulating an attacker with 14 kbps per bot when ASes have resources following the step model.

When comparing Figures 4 and 5, we see that a larger attack traffic generation capability simply changes the size of the botnet needed to have a given impact. For example, an attacker needs over 3 million bots that can generate 14 kbps to attack the top ten ASes, but only 400,000 bots are necessary if each can generate 128 kbps. At the same time, the number of congested collateral ASes for a given destructiveness is the same. To achieve a destructiveness of 1, an attacker under the CodeRed or GT-DDoS distributions congests 6 or 71 collateral ASes, respectively.

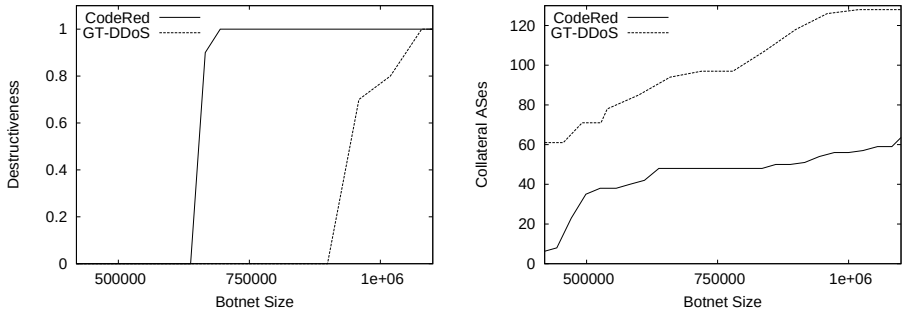
When the resources for the target ASes are doubled (see Figure 6), the advantage of having botnets spread through more ASes is more pronounced. The CodeRed distribution has bots distributed over 4746 different ASes versus the GT-DDoS distribution with only 720 ASes. With traffic coming from more directions and greater chance of traffic traversing the target link, an attacker with the CodeRed distribution can achieve a destructiveness of 1 with only 700,000



(a) Destructiveness (b) Congested Collateral ASes

Fig. 5. Results when simulating an attacker with 128 kbps per bot when ASes have step based resources.

bots and 48 congested collateral ASes. To achieve the same destructiveness, an attacker with the GT-DDoS distribution needs an additional 308,000 bots, and congests 128 collateral ASes.



(a) Destructiveness (b) Congested Collateral ASes

Fig. 6. Results when the top ten ASes double their resources. (attacker traffic generation = 128 kbps per bot)

These results indicate that an attacker with a realistically distributed botnet under realistic traffic and network settings can launch a focused Coremelt attack which causes core links to fail. This attacker can launch such an attack without raising suspicion by congesting few tributary links.

5 Previous Work and Potential Coremelt Defenses

In this section, we discuss work related to attacks on the core of the Internet or DoS defenses. We also discuss if such DoS defenses could mitigate a Coremelt attack.

Magoni [1] analyzes attacks on the core of the Internet. His study shows how the targeted removal of links could significantly impact connectivity in the Internet. However, his paper simply assumes that a malicious party could disable a link, without discussing any specific attack mechanism.

A number of prior works examine how to prevent DoS attacks using systems to trace traffic to the source, capabilities that allow legitimate traffic preference over attack traffic, puzzles to force attackers to expend work to impact the victim, or techniques to balance resource allocation across different users. Unfortunately, none of these solutions provides a satisfactory solution to the Coremelt attack, because these defense mechanisms attempt to stop traffic that is unwanted by the destination or use a definition of fairness that fails to protect non-attack traffic in the worst case scenario.

Trace back systems [5–10] help defend against DoS attacks where an attacker would use a small number of machines from the same network to flood a victim with traffic containing spoofed addresses. Once the victim knows the source of the traffic, administrators on the attacker's network can turn off ports, stopping the attack traffic. In Coremelt and other DDoS attacks, a victim has trouble separating legitimate traffic from attack traffic. The flows between bots in the Coremelt attack consume relatively limited bandwidth and appear as legitimate as any other flow traversing the core link. Without a way to differentiate legitimate and attack traffic, tracing traffic provides no help during a Coremelt attack.

In capability-based systems [11, 12], traffic which a destination wants to receive is given priority at congested routers. The destination gives legitimate sources a capability that ensures prioritized delivery. If an attack occurs, attack traffic will lack the proper capability and be dropped by congested routers. In a variant of capability-based systems [13], rather than approving wanted traffic the destination asks the source's ISP to filter unwanted traffic. In Coremelt, bots want traffic from other bots and will grant capabilities for the traffic (or never mark attack traffic as unwanted), easily circumventing capability-based DoS defenses.

One solution to DoS attacks is to use puzzles to increase the cost for an attacker to consume victims' resources [14–18]. If the amount of work needed to complete the puzzle is large enough, the attacker will be unable to launch a successful attack. Most of these are designed as challenges a client must perform before a server will provide a service. However, Portcullis [17] uses puzzles with a connection's initial packet to allow clients to acquire capabilities in a capability-based DoS system. After acquiring the capability, the legitimate traffic requires no additional work and can proceed unhampered by the DoS attack. If we were to adopt puzzles for every packet, the puzzles may become the bottleneck rather than the links. During a Coremelt attack, the target link will increase the re-

sources senders need to invest to complete puzzles, effectively degrading the performance of any machine using the target link.

One final approach to DoS mitigation is to fairly distribute the available resources across all users [19, 20]. In these schemes, max-min fair bandwidth allocation ensures all flows achieve the same output rate.[7] The goal is to isolate legitimate traffic from attack traffic such that an attack flow can only use as much bandwidth as a non-attack flow. How flows are defined plays a key role on how a Coremelt attack impacts legitimate traffic. In Core-Stateless Fair Queueing [20], the endpoints of a connection define a flow (i.e., IP addresses of the client and the server). With a small number of attackers flooding a given link, the fair sharing will prevent the attack flows from impacting legitimate users. However, in a Coremelt attack with N bots, there are $O(N^2)$ source-destination pairs contributing bandwidth to the link. With so many pairs, even if bandwidth is shared fairly (i.e., every flow or source-destination pair receives the same amount of bandwidth), the bandwidth a legitimate flow receives is drastically reduced. Chou et al. [19] focus on fair allocation of bandwidth within the core of the network and define flows based on the source and destination router (i.e., where a packet enters and exits the core of the network). With flows defined by routers—rather than endpoints—a botnet must be widely distributed to disrupt all traffic across the link. When legitimate traffic traverses the same pair of routers as attack traffic, the bandwidth allocation mechanism considers all of the traffic the same flow. As a result, once the link is congested, the scheme will drop packets from this flow with no preferential treatment for non-attack traffic. However, traffic traversing pairs of routers that include zero Coremelt traffic will proceed unhampered, independent of the amount of attack traffic.

6 Conclusion

Internet connectivity is crucial for social, economic, and government purposes. Loss of connectivity due to malicious activity can cause serious financial and physical damage to services. Traditional Denial of Service (DoS) attacks attempting to disrupt connectivity flood a victim with unwanted traffic. Researchers have proposed a number of defenses to address such DoS attacks. In this work, we present Coremelt, a new type of DoS attack where N attackers send traffic to each other, overloading the core of the network with their $O(N^2)$ pairwise connections. The malicious sources and destinations want the traffic, allowing the packets to elude traditional DoS defenses that assume attack traffic is unwanted by the receiver. Simulation of the attack on a simplified model of the Internet topology with a realistic attacker model shows that a Coremelt attack can cause serious congestion in the Internet. Hopefully, this work will motivate researchers to investigate solutions to this debilitating attack.

[7] In addition to equal sharing of bandwidth, network administrators can assign different weights to different flows. Flows with larger weights will receive a larger, but fixed, fraction of the bandwidth.

7 Acknowledgments

We would like to thank Chris Lee and Wenke Lee for sharing their data on real botnets. We would also like to thank the anonymous reviewers for their insightful comments and feedback that helped improve the quality of this paper.

References

1. Magoni, D.: Tearing down the internet (2003)
2. Savage, S., Cardwell, N., Wetherall, D., Anderson, T.: TCP congestion control with a misbehaving receiver. ACM SIGCOMM Computer Communication Review **29**(5) (1999)
3. CAIDA: As relationships dataset - jan. 5, 2009. `http://www.caida.org/data/active/as-relationships/`
4. Moore, D., Shannon, C.: The caida dataset on the code-red worms - july and august 2001. `http://www.caida.org/data/passive/codered_worms_dataset.xml`
5. Burch, H., Cheswick, B.: Tracing anonymous packets to their approximate source. In: Proceedings of the Large Installation System Administration Conference. (2000)
6. Goodrich, M.: Efficient packet marking for large-scale IP traceback. In: Proceedings of ACM CCS. (November 2001)
7. Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Kent, S.T., Strayer, W.T.: Hash-based IP traceback. In: Proceedings of ACM SIGCOMM 2001. (August 2001) 3–14
8. Snoeren, A.C., Partridge, C., Sanchez, L.A., Jones, C.E., Tchakountio, F., Schwartz, B., Kent, S.T., Strayer, W.T.: Single-packet IP traceback. IEEE/ACM Transactions on Networking (ToN) **10**(6) (December 2002)
9. Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Practical network support for IP traceback. In: Proceedings of ACM SIGCOMM. (August 2000)
10. Yaar, A., Perrig, A., Song, D.: Pi: A path identification mechanism to defend against DDoS attacks. In: Proceedings of IEEE Symposium on Security and Privacy. (May 2003)
11. Yaar, A., Perrig, A., Song, D.: SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks. In: Proceedings of IEEE Symposium on Security and Privacy. (May 2004)
12. Yang, X., Wetherall, D., Anderson, T.: A DoS-limiting network architecture. In: Proceedings of ACM SIGCOMM. (August 2005)
13. Argyraki, K., Cheriton, D.: Scalable Network-layer Defense Against Internet Bandwidth-Flooding Attacks. IEEE/ACM Transactions on Networking (2009)
14. Aura, T., Nikander, P., Leiwo, J.: DoS-resistant authentication with client puzzles. In: Proceedings of Security Protocols Workshop. (2001)
15. Dean, D., Stubblefield, A.: Using client puzzles to protect TLS. In: Proceedings of USENIX Security Symposium. (2001)
16. Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: Proceedings of ISOC NDSS. (1999)
17. Parno, B., Wendlandt, D., Shi, E., Perrig, A., Maggs, B., Hu, Y.C.: Portcullis: Protecting connection setup from denial-of-capability attacks. In: Proceedings of the ACM SIGCOMM. (August 2007)
18. Wang, X., Reiter, M.: Defending against denial-of-service attacks with puzzle auctions. In: Proceedings of IEEE Symposium on Security and Privacy. (May 2003)

19. Chou, J., Lin, B., Sen, S., Spatscheck, O.: Proactive surge protection: A defense mechanism for bandwidth-based attacks. In: USENIX Security Symposium. (2008)
20. Stoica, I., Shenker, S., Zhang, H.: Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In: Proceedings of ACM SIGCOMM. (1998)