# SECURING APACHE : DOS & DDOS ATTACKS - I

*In this part of the series, we focus on DoS/DDoS attacks, which have been among the major threats to Web servers since the beginning of the Web 2.0 era.*

## Denial of Service (DoS) attacks

The goal of a Denial of Service (DoS) attack is to disrupt some legitimate activity, such as browsing Web pages, email functionality or the transfer of money from your bank account. It could even shutdown the whole Web server. This denial-of-service effect is achieved by sending messages to the target machine such that the "message" interferes with its operation and makes it hang, crash, reboot, or do useless work.

In a majority of cases, the attacker's aim is to deprive clients of desired server functionality.
One way to interfere with legitimate operations is to exploit vulnerabilities on the target machine or application, by sending specially crafted requests targeting the given vulnerability (usually done with tools like Metasploit). Another way is to send a vast number of messages, which consume some key resource of the target machine, such as bandwidth, CPU time, memory, etc. The target application, machine, or network spends all of its critical resources on handling the attack traffic, and cannot attend to legitimate clients.

Of course, to generate such a vast number of requests, the attacker must possess a very powerful machine — with a sufficiently fast processor — and a lot of available

network bandwidth. For the attack to be successful, it has to overload the target's resources. This means that an attacker's machine must be able to generate more traffic than a target, or its network infrastructure, can handle.

## Attack scenario

Here is a simple scenario: an attacker sends a large number of requests to a Web server — for example, a website that hosts HD image files at a particular URL, say`www.example.com/images/HD_images.html`. Let's also assume that this page contains about 50-60 images. Now, every time a user reloads this page, it consumes a large portion of the Web server's bandwidth. Now, here, an attacker could design a separate HTML page, with an `iframe`embedded in it, like what's shown below:

```
<html>
    <iframe src=http://www.example.com/images/HD_images.html
width=2 height=2></iframe>
</html>
```

Let's suppose that instead of a single `iframe`, the attacker copies and pastes the above code 1,000 times in the same page, and also adds a `meta` refresh tag as follows:

```
<html>
    <head>
        <meta http-equiv="refresh" content="2">
    </head>
    <iframe src=http://www.example.com/images/HD_images.html
width=2 height=2></iframe>
    <iframe src=http://www.example.com/images/HD_images.html
width=2 height=2></iframe>
    :
    :
    : (1000 times)
</html>
```

Such a page, when loaded, will send the same request 1,000 times every 2 seconds, and will consume a lot of the Web server's bandwidth. Thus, the target server will not be able to respond to other clients, and eventually, legitimate clients will be denied services from the server.

Now let us assume that an attacker would like to launch a DoS attack on `example.com` by bombarding it with numerous messages. Also assume that `example.com` has abundant resources and considerable bandwidth (which is most often the case). It is then difficult for the attackers to generate a sufficient number of messages from a single machine (as in the above scenario) to overload those resources.
However, imagine the consequences if they got 100,000 machines under their control, in order to simultaneously generate requests to `example.com`. Each of the attacking machines (compromised machines that have been infected by malicious code) may be only moderately provisioned (have a slow processor and be on a mere modem link), but together, they form a formidable attack network — which, with proper use, could overwhelm even a well-provisioned victim site. This is a distributed denial-of-service (DDoS) attack, and the machines under the attacker's control are termed as zombies/agents.

# Distributed denial-of-service (DDoS) attacks

In a typical DDoS attack, the attacker's "army" consists of master zombies and slave zombies. The attacker coordinates and orders master zombies and they, in turn, coordinate and trigger slave zombies. More specifically, the attacker sends an attack command to the master zombies, and activates all attack processes on those machines, which are in hibernation, waiting for the appropriate command to wake up and start attacking.

Then the master zombies duplicate the attack commands to each of their slave zombies, ordering them to mount a DDoS attack against the victim. In this way, the

zombie systems begin to send a large volume of packets to the victim, flooding it with useless loads, and exhausting its resources.

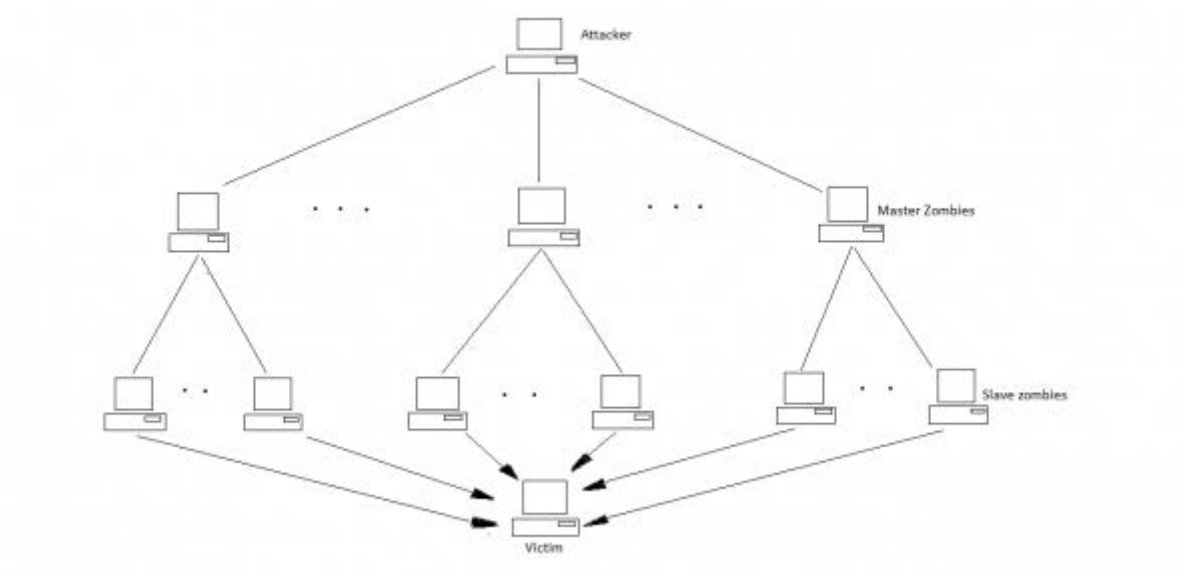


Figure 1: A typical DDoS attack

In DDoS attacks, spoofed source IP addresses are used in the packets of the attack traffic. Attackers prefer to use such counterfeit source IP addresses for two major reasons: first, to hide the identity of the zombies, so that the victim cannot trace the attack back to them. The second reason is to discourage any attempt by the victim to filter out the malicious traffic.

# Types of DoS attacks

Though there are numerous flavours of DoS attacks, here we'll discuss only those which affect the Web server and Web applications.

## TCP SYN flooding attacks

DoS attacks often exploit stateful network protocols, because these protocols consume resources to maintain state. TCP SYN flooding is one such attack, and had a wide impact on many systems. When a client attempts to establish a TCP

connection to a server, the client first sends a SYN message to the server. The server acknowledges this by sending a SYN-ACK message to the client. The client completes establishing of the connection by responding with an ACK message. The connection between the client and the server is then open, and service-specific data can be exchanged between them.

The abuse occurs at the “half-open” state when the server is waiting for the client’s ACK message, after sending the SYN-ACK message to the client. The server needs to allocate memory to store information about the half-open connection, and this memory will not be released until the server either receives the final ACK message, or the half-open connection expires (times out).

Attackers can easily create half-open connections by spoofing source IPs in SYN messages, or ignoring SYN-ACKs. The consequence is that the final ACK message will never be sent to the victim. Because the victim normally only allocates a limited amount of space in its process table, too many half-open connections will soon fill the space.

Even though the half-open connections will eventually expire due to their timeout, zombies can aggressively send spoofed TCP SYN packets, requesting connections at a much higher rate than the expiration rate. Finally, the victim will be unable to accept any new incoming connections, and thus cannot provide services.

## ICMP Smurf attacks

The Internet Control Message Protocol (ICMP) is used to handle errors and exchange control messages. ICMP can be used to determine if a machine on the Internet is responding. To do this, an ICMP echo request packet is sent to a host. If a host receives the packet, that host will return an ICMP echo reply packet. A common implementation of this process is the `ping`application.

In this attack, spoofed IP packets containing ICMP echo requests, with a source address like that of the target system, and a broadcast destination address, are sent to the intermediate network. (Broadcast addresses are specially allocated addresses within all network subnets, used to broadcast messages to the whole network. All hosts within a given subnet receive packets sent to these broadcast addresses and in some cases — ICMP protocol, for instance — respond to them.)

Sending the ICMP echo request to a broadcast address triggers all hosts in the network to respond with an ICMP response packet to the spoofed address (the victim's), thus creating a large mass of packets which are routed to the victim's address. Networks may include up to hundreds of hosts; hence, one attack echo request results in hundreds of packets flooding the victim's site.

## UDP flooding attacks

By patching or redesigning the implementation of TCP and ICMP protocols, current networks and systems have incorporated new security features to prevent TCP and ICMP attacks. Nevertheless, attackers may simply send a large amount of UDP packets towards a victim. Since an intermediate network can deliver higher volumes of traffic than the victim network can handle, the flooding traffic can exhaust the victim's connection resources.

Pure flooding can be done with any type of packets. Attackers can also choose to flood service requests so that the victim cannot handle all requests with its constrained resources (i.e., service memory or CPU cycles). UDP flooding is similar to "flash crowds" that occur when a large number of users try to access the same server simultaneously.

## Distributed Reflected Denial of Service (DRDoS) attacks

Unlike typical DDoS attacks, in the case of DRDoS attacks, the army of the attacker consists of master zombies, slave zombies, and reflectors. This resembles typical

DDoS attacks up to a particular point. The attackers have control over master zombies, which, in turn, have control over slave zombies.

The difference in this type of attack is that slave zombies are led by master zombies to send a stream of packets with the victim's IP address as the source IP address, to other, uninfected machines (known as *reflectors*), causing these machines to connect to the victim. Thus, the reflectors send the victim a greater volume of traffic, because they see the victim as requesting it. Therefore, in DRDoS attacks, the attack is mounted by non-compromised machines, which mount the attack without being aware of the hostile intent.

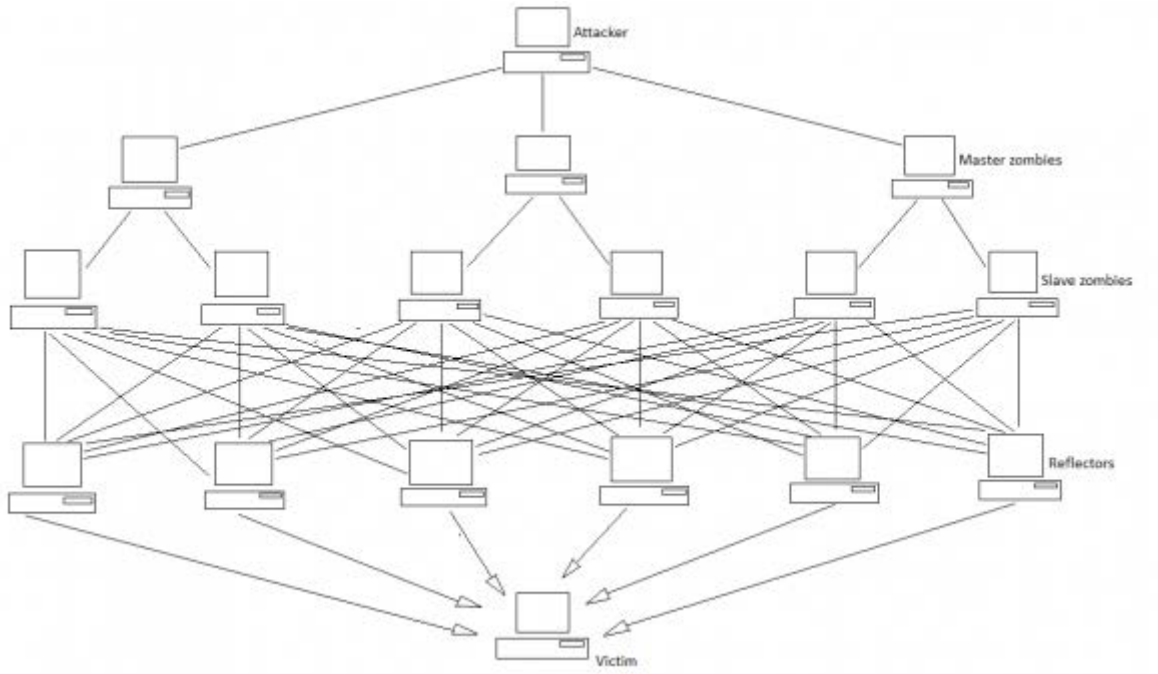


Figure 2: A typical DRDoS attack

Comparing the two scenarios of DDoS attacks, we should note that a DRDoS attack is more detrimental than a typical DDoS attack, because a DRDoS attack has more machines to share the attack — the attack is more distributed and so creates a greater volume of traffic.

Source : http://www.opensourceforu.com/2011/04/securing-apache-part-8-dos-ddos-attacks/