

THE "TRIBE FLOOD NETWORK 3000" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

"Tribe Flood Network 3000": A theoretical review
of what exactly Distributed DOS tools are, how they can be used,
what more dangerous features can be implemented in the future, and starting
points on establishing Network Intrusion Detection Rules for DDOS.

Many technically uninformed people consider DDOS as a weapon, that should not
be publicly evolved and distributed. This is the only further thing I'll be
releasing to explain DDOS tools, comprehensible for EVERYONE, and future
features that may be implemented in DDOS: a brief theoretical description.
BTW: People with technical knowledge may skip over the most stuff in I. and II.

- I. What is distributed DOS, what can it be used for, how does it operate?
- II. What are DDOS features, what are future DDOS features, how is DDOS evolving?
- III. DDOS, an exploit or not? Should it be published? What is the main problem?
- IV. How can DDOS traffic be detected by Network Intrusion Detection (NIDS)?

I.

What is distributed DOS?

Distributed DOS, like any distributed concept, will make it easy to coordinate
a procedure that is launched from many computers. In this case it is Denial
Of Service in form of packet flooding, to overload network links.

DDOS IS NOT A HACKING TOOL CATEGORY. Distributed DOS tools are PENETRATION
tools. They do not exploit security vulnerabilities, but they can demonstrate
what amount of traffic a host can or cannot handle. Distributed DOS has been
used a long time by professional security consultants for penetration testing.
Before there were DDOS attack tools, there have been commercial, non-open-source
programs out that could launch distributed packet floods. Those were used in the
information security consulting business, to perform a security service called
"Capacity Management". The purpose of Capacity Management is to determine how
much traffic a network can handle, to see if the targets bandwidth has to be
improved, or if it can handle enough traffic while providing service reliably.

What can it be used for?

It can overload, or flood if you want, network links. It sends meaningless
packets, the overall amount of data being more than the network can process.
The impact is that the targets can not be reached over a network. That is all.

How does it operate?

The basic concept is that you install a huge amount of DOS servers on different
hosts. They will await commands from a central client. A central client can then
message all the servers, and instruct them to send as many traffic as they can
to one target. The tool distributes the work of flooding a target amongst all
available DOS servers, therefore it is called a distributed concept.

Before these tools were available, an attacker (or penetration tester) would
have to telnet into all the hosts that he wanted to use, log in as a user,
and manually launch a command to flood a target on each of the hosts that
should flood, for example using the UNIX standard tool ping: 'ping -f target'

THE "TRIBE FLOOD NETWORK 3000" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

II.

What are DDOS features?

The actual attack tools don't do simple flooding, but variations of it which involves using actual weaknesses in a protocol to a) make an attack more powerful b) make an attack harder to track back. First, current DDOS tools spoof source addresses. They are sending raw IP packets, and due to the nature of the internet protocol the source addresses can be fake ones, and single (not connection oriented) packets will still reach their destination. This is basically what makes backtracking of the attacks so hard. DDOS is also exploiting protocol weaknesses, it for example can open up half-open TCP connections by SYN flooding. This is a very old and well known protocol vulnerability, and feasible countermeasures are present. To make attacks more powerful, DDOS can generally use any protocol vulnerability that can be exploited by sending single, not connection oriented packet traffic to a host.

What are future DDOS features?

Things that can still be implemented, but have not in publicized tools, are protocol vulnerabilities as mentioned above. One of those is the "stream" attack (discovered by Tim Yardley, stream.c and spank.c demonstrate the vulnerability and are public). Stream attack sends TCP packets with either ACK or both SYN and ACK flags set. Because they are not part of a connection, they will "confuse" a target machine and take some time to be processed by the operating system. If this attack is used in a distributed way, the attacker can overload machines with less hosts. From what I've heard, distributed stream attack IS already implemented in private DDOS tools. It is very trivial to implement this feature. Possibility 2 that is not implemented yet are multicast addresses. Multicast addresses are routed (forwarded) specially by routers, they can multiply one packet into several ones. The concept would be to send out packets with a multicast (224.x.x.x) source. A target could send an error message back to multicast destinations, and multiply the bandwidth. This concept has also been mentioned by Tim Yardley. Another concept could be to purposefully send special strings in the flood traffic, strings that Intrusion Detection Systems (IDS) could falsely interpret as break-in attempts, the impact would be false alarms and affected IDS could get overloaded or crash.

How is DDOS evolving?

As I mentioned, the first tools that did distributed denial of service were commercial penetration tools. The origin of using general DOS is certainly IRC (Internet Relay Chat), where kiddies can take over control of channels if they temporarily take out computer systems with DOS. The first packet flooding DOS that involved multiple servers flooding was "smurf". Smurfing relied on mis-configured networks replying back to a broadcast address, sending one packet would result in hundreds bouncing back. Then, most of those networks were fixed, and attackers compromised a lot of hosts, preferably hosts with high bandwidth, and started flooding manually from them. Because this took a lot of time, attackers wrote servers which they installed on the hosts they had compromised. They no longer needed to log in, but only message those servers. The DDOS attack tools I know of are, in chronological release order: fapi (private, by ?), blitznet (public, by phreeon), trinoo (private, by phifli), TFN (public, by me), stacheldraht (private, by randomizer), shaft (private, by ?), TFN2K (public, by me), Trank (TRinoo + spANK.c?, private).

THE "TRIBE FLOOD NETWORK 3000" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

The recent development has also continued in other ways, since people were monitoring traffic for very DDOS-program-specific traffic (like known character strings, known passwords, default ports), there have been many small variations made to the code of the above tools, by attackers, to prevent being detected.

III.

DDOS, an exploit or not?

No. DDOS itself is not an exploit. It just makes an existing concept more easy. Take the distributed.net RC5 challenge and distributed password crackers. They are not exploits. But they are exposing a weakness, that many passwords can be brute forced faster than people think. DDOS shows that many networks are not as strong as they seem to be and can be overloaded faster than people used to think. Additionally, there are actual exploits implemented in DDOS exploits, that exploit security holes in network protocols currently used on the Internet. These security holes must not necessarily be exploited to make DDOS possible, but they do make the impact of DDOS attacks more powerful. Such exploits are the possibility of arbitrarily spoofing IP addresses, SYN flooding, IP stack attacks with bad fragmentation, header offsets and other "magic packets", the stream vulnerability, and missing authentication and security of traffic known as connection-less or stateless.

Should it be published?

That is for you to decide. It is your personal opinion. But people will continue to publish vulnerabilities. Hundreds of talented security analysts are professionally researching vulnerabilities in software, and posting exploit programs, which can often be used to instantly compromise a system running the vulnerable software at root level. The past has shown, that since security vulnerabilities were a problem on the internet, people have been ignoring advisories containing only the information THAT something was vulnerable to an attack, disregarding them as being "completely theoretic". Only when people wrote up and posted ready-to-(ab)use vulnerability exploits, the severity of vulnerabilities became clear, and people would make an effort to counter those vulnerabilities.

What is the main problem?

The main problem, that made attacks against sites as big as yahoo.com possible, is the bad overall security on the internet. With ONLY a DDOS tool in his hands, Joe Attacker cannot do anything. But security vulnerabilities are omni-present on the majority of hosts on the net. An awful lot of these hosts are not caring about their security, as a result they are running software that is KNOWN to be vulnerable, and against which public exploit programs exist. An attacker has only to run one of the public exploit programs and he is granted full access to such hosts. And various people have been able to compromise THOUSANDS of hosts with well-known, old vulnerabilities. Even high speed university networks, which originally built the foundation of internet architecture have proven to be insecure. With full control over thousands of hosts, it is easy to concentrate all of these hosts resources, and to be able to attack almost anything on the internet.

IV.

THE "TRIBE FLOOD NETWORK 3000" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

How can DDOS traffic be detected by Network Intrusion Detection (NIDS)?

The mistake everyone has been making is to search for default strings of special DDOS tools, for default values, ports, passwords, etc.

To establish Network Intrusion Detection capability in order to spot these tools, that operate via connectionless raw packets, people will have to start looking for general signs of DDOS traffic, signs that are obvious and traffic that is extensively anomalous and suspicious.

There are two kinds of DDOS-generated traffic, control traffic (between DDOS client and servers) and flood traffic (between DDOS servers and DDOS victim). Credits to rain forest puppy, Dave Dittrich, and Axent Security Team for providing some initial hints I needed to write this up.

Anomaly 0: This is not real "DDOS" traffic, but it can be a viable method of determining the origin of DDOS attacks. As observed by RFP, an attacker will have to resolve his victim's hostname before a DDOS attack. BIND name servers are capable of recording these requests. You can either send them a WINCH signal with 'kill', or you can specify query logging in the BIND configuration. A single PTR type query before an attack indicates the request was made from the attackers host, a great load of PTR type query for a DDOS victim before an attack indicates that the flood servers have been fed a host name and each server was resolving the hostname for itself.

Anomaly 1: Amount of bandwidth exceeds a maximum threshold that is expected normal traffic for a site could cause. Alternatively, the threshold can be measures in the amount of different source addresses in the traffic. These are clear signs of flood traffic and ACL rules can be implemented on the backbone routers that detect these signs and filter traffic.

Anomaly 2: Oversized ICMP and UDP packets. Stateful UDP sessions are normally using small UDP packets, having a payload of not more than 10 bytes. Normal ICMP messages don't exceed 64 to 128 bytes. Packets that are reasonably bigger are suspicious of containing control traffic, mostly the encrypted target(s) and other options for the DDOS server. Once (non-decoy) control traffic is spotted, one of the DDOS servers' location is revealed, as the destination IP address is not spoofed in control traffic.

Anomaly 3: TCP packets (and UDP packets) that are not part of a connection. The stealthiest DDOS tools use random protocols, including connection-oriented protocols, to send data over non-connection-oriented channels. Using stateful firewalls or link-state routing can discover these packets. Additionally, packets that indicate connection requests with destination ports above 1024, with which no known service is registered and running, are highly suspicious.

Anomaly 4: Packet payload contains ONLY alphanumeric character (e.g. no spaces, punctuation, control characters). This can be a sign that the packet payload is BASE64-encoded, and therefore contains only base64 characters. TFN2K is sending such packets in its control traffic. A TFN2K (and TFN2K derivatives) specific pattern is a string of repeating A's (AAAA...) in the payload, since the buffer size is padded by the encryption routine. If the BASE64 encoding is not used, and the payload contains binary encrypted traffic, the A's will be trailing binary \0's.

THE "TRIBE FLOOD NETWORK 3000" DISTRIBUTED DENIAL OF SERVICE ATTACK TOOL

Anomaly 5: Packet payload contains ONLY binary, high-bit characters. While this can be a binary file transfer (traffic transmitted over ports 20, 21, 80, etc. must be excluded if this rule is applied), especially if contained in packets that are not part of valid stateful traffic, it is suspicious of being non-base64 encoded, but encrypted control traffic that is being transmitted in the packet payload.

- Mixer