UCRL-ID-136939 Rev. 1

Distributed Denial of Service Tools Trinoo, Tribe Flood Network, Tribe Flood Network 2000, and Stacheldraht

P.J. Criscuolo

February 14, 2000

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

Disclaimer

The information within this paper may change without notice. Use of this information constitutes acceptance for use in an AS IS condition. There are NO warranties with regard to this information. In no event shall the author be liable for any damages whatsoever arising out of or in connection with the use or spread of this information. Any use of this information is at the user's own risk.

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48 between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL. The rights of the Federal Government are reserved under Contract 48 subject to the restrictions agreed upon by the DOE and University as allowed under DOE Acquisition Letter 97-1.

Overview

One type of attack on computer systems is known as a Denial of Service (DoS) attack. A Denial of Service attack is designed to prevent legitimate users from using a system. Traditional Denial of Service attacks are done by exploiting a buffer overflow, exhausting system resources, or exploiting a system bug that results in a system that is no longer functional. In the summer of 1999, a new breed of attack has been developed called Distributed Denial of Service (DDoS) attack. Several educational and high capacity commercial sites have been affected by these Distributed Denial of Service attacks. A Distributed Denial of Service attack uses multiple machines operating in concert to attack a network or site. There is very little that can be done if you are the target of a DDoS. The nature of these attacks cause so much extra network traffic that it is difficult for legitimate traffic to reach your site while blocking the forged attacking packets. The intent of this paper is to help sites not be involved in a DDoS attack.

The first tools developed to perpetrate the DDoS attack were Trin00 and Tribe Flood Network (TFN). They spawned the next generation of tools called Tribe Flood Network 2000 (TFN2K) and Stacheldraht (German for Barb Wire). These Distributed Denial of Service attack tools are designed to bring one or more sites down by flooding the victim with large amounts of network traffic originating at multiple locations and remotely controlled by a single client.

This paper discusses how these DDoS tools work, how to detect them, and specific technical information on each individual tool. It is written with the system administrator in mind. It assumes that the reader has basic knowledge of the TCP/IP Protocol.

Description

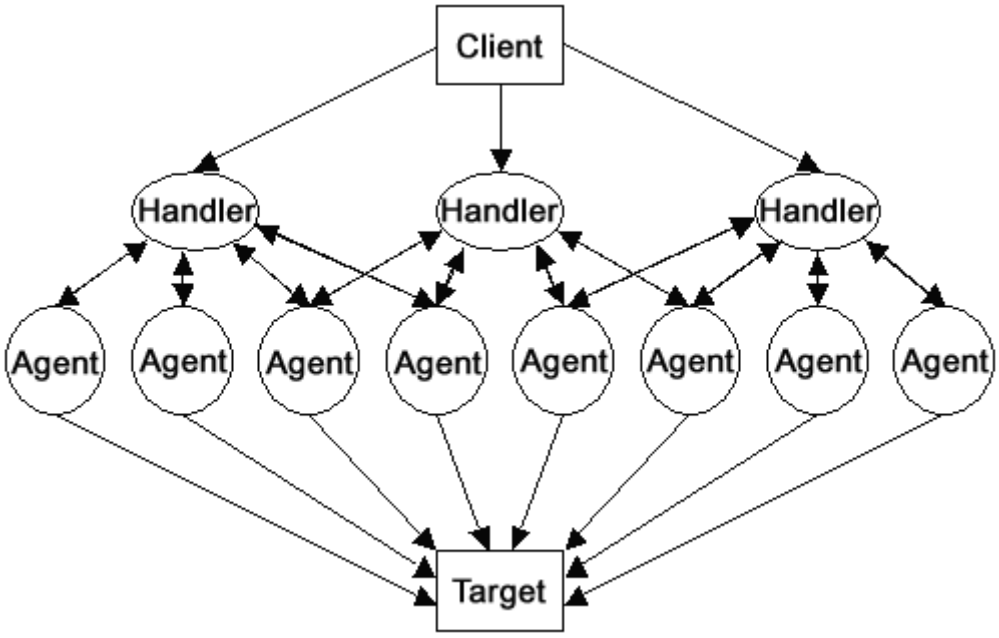
Each of these DDoS tools is based on the same premise and topology. These tools are not used to capture data or infiltrate a computer system. They are used to disrupt the normal network traffic to a host. These DDoS attacks are extremely difficult to trace because of the stealth capabilities built into the tools. They utilize encryption to hide their communications and forge their source addresses to hide their location. These tools are controlled remotely. If the target site is able to trace the forged source addresses to the agents, it is still necessary to trace back to the handlers, and then finally to the client, or attacker.

These DDoS attacks are a two-phase assault. The attacker will spend a large amount of time preparing for the first phase of an attack. This phase of the assault involves compromising as many systems as possible. The attacker needs multiple computers to generate the large numbers of packets needed to shut down a system. All of these attack tools run on both Linux and Solaris while TFN2K runs on Windows as well as Linux and Solaris. The attacker must find multiple computer systems with weak security to infiltrate and install the DDoS tools, as well as paraphernalia such as a rootkit, to hide the DDoS's presence. To prevent your system from being the source of a DDoS attack, it is important that you keep your computer systems secure and up-to-date with the most recent security patches. The second phase of this attack cannot take place until phase one

is successful in finding enough machines to generate the extremely high volume of traffic.

The second phase is the actual Denial of Service attack. The compromised systems will generate the network traffic to bring down a targeted site. These compromised systems are considered secondary victims of the Denial of Service attack. The network traffic that is generated exploits the TCP Protocol. The target of the flood cannot handle the packets that are sent in an efficient manner and will cause resources to be allocated. Many times it would not matter how the target site interprets the packets. The volume of packets is so great the network itself becomes congested with artificial traffic. This congestion does not allow legitimate traffic to pass.

The topology of the Distributed Denial of Service attack consists of four parts. The compromised systems are broken down into handlers and agents. The agents are where the disabling network traffic is generated. One or more handlers control these agents. The handlers maintain a list of all responding agents. The handlers signal the agents when to begin an attack and specify the method of attack. The attacker, or client, controls one or more handlers and each agent can respond to more than one handler.



Communication between the handlers and agents in the earlier two tools, Trin00 and TFN, was done using the TCP/IP protocol. These communications use no encryption, except for a rudimentary password encryption used to start sending commands. The latest tools, TFN2K and Stacheldraht, encrypt most of the communication between the handler and agents using either the Blowfish or CAST encryption algorithms.

The method of attack varies with these tools. Trin00 only attacks with a UDP (User Datagram Protocol) flood. All the other tools allow the attacker to choose between a

UDP flood, SYN flood, and ICMP (Internet Control Message Protocol) echo reply type flood. TFN2K added the targa3 attack and a mixed attack that will use the UDP, SYN and ICMP floods in a 1:1:1 ratio. See Appendix E – Flood Attacks Described for detail descriptions on each of these forms of attacks.

Detection and Prevention

The most important aspect of these distributed attacks is that the attacker needs compromised computer systems to carry out the attack. If the Internet, as a community, were to make sure that each of its subnets were secure, there would be no place for the hackers to place their tools. This includes making sure all systems are secure and fully patched and unneeded services are turned off. To enhance computer security, enforce the use of strong password rules by all users since hackers use weak passwords to gain unauthorized access.

One of the stealth techniques that these tools use is to forge the source IP address in the header of the IP packets. A forged source address prevents the target site from knowing where that attack is coming from. Routers can be configured so that packets will not route if their source address is not from within the subnet served by the router. This would not stop all of the packets from getting out, but would allow them to be traced to the attack machine.

There are presently three tools on the Internet that will help you discover if the handlers and agents are on your system. The first is by the National Infrastructure Protection Center (NIPC) called find_ddosv31. It runs on Solaris version 2.5.1, 2.6, and 7 for the Sparc and Intel platforms as well as Linux on Intel platforms.

Version 3.1 of the tool detects TFN2K client, TFN2K agent, Trin00 agent, Trin00 handler, TFN agent, TFN client, Stacheldraht handler, Stacheldraht client, Stacheldraht demon and TFN-rush client. It detects these agents and handlers by searching the hard drive for known strings in the binary of the attack tools. It is a program that needs to be run locally on each host to detect the presence of the attack tools. CIAC had much success detecting the attack tools on Solaris operating systems. We had a difficult time getting it compiled on Linux Red Hat 6.1. One drawback of this tool is that the hacker community is now compressing the binaries. This allows the binary to still execute but the strings are hidden in a compressed binary. The NIPC tool can be downloaded from http://www.fbi.gov/nipc/trinoo.htm.

David Dittrich of the University of Washington has developed a tool called ddos_scan. It scans for the Trin00 agent, TFN agent, and Stacheldraht agent. It does **not** detect TFN2K at this time. The tool works by scanning the network with handler-agent communication packets and then watches the return packets for certain strings. This utility scans a complete subnet from a single node on that subnet. If the attack tools source code were modified to accept communications from a different port or the default passwords were changed, then this tool would **not** be successful. It can be found at http://staff.washington.edu/dittrich/misc/ddos_scan.tar.

David Brumley at Stanford University wrote a remote detector for Trin00 agent, TFN agent, and Stacheldraht agent called rid. It also looks for the default ports and passwords used by these attack tools. Rid searches an entire subnet from a single node as well as searches hosts from a list. Rid also uses a configuration file to change the ports and strings it looks for and the hosts it scans. This file can be modified easily in the event an attack tool is discovered by other means. The ports and passwords can be entered into the configuration file, adding to the search list. CIAC has had difficulty getting this scanner to run under Red Hat Linux 6.1, but have seen good results on Solaris 2.7. It can be found at http://packetstorm.securify.com/distributed/rid-1_0.tgz.

Recommendations

To keep your systems from being used to attack others make sure that all systems are fully patched with the latest released information from the vendor. Make sure a good password policy is in place on every machine in your network and use a one-time-password scheme or encryption to prevent an intruder who gains access to one machine on a network from installing a sniffer and capturing information to access the rest of the network. Turn off all unneeded services. Check with vendors on a regular basis for updates and patches. Hackers are constantly developing new tools and discovering exploits. As new technology is introduced, so is the availability for new exploits, so audit systems on a regular basis. Make sure all servers and routers are logging everything possible.

David Brumley's tool, rid, is the most promising at this time. It is identical to Dave Dittrich's program in how it discovers the agents. CIAC recommends rid's use over dds because of the ability to add to the search parameters through the configuration file. Most users will be able to change this file, while with dds, modification to the source code is the only way the search parameters can be changed. NIPC's tool is very successful in detecting the binaries for all of the attack tools on Solaris.

Firewalls should be installed on the outer edge of networks. These firewalls can be configured to filter and log incoming and outgoing traffic. They will also allow you to prevent certain protocols from entering or leaving a network. For example, all ICMP could be prevented from entering or leaving a network, but all ICMP traffic will flow normally behind the firewall for diagnostic purposes. This eliminates many of the tools these agents use to communicate and attack.

Routers should be configured such that all outgoing traffic is checked to make sure that the packet source IP belongs to the subnet that the router services. This is called Network Ingress Filtering. RFC 2267 discusses how this can be used in limiting the effectiveness and scope of a DDoS. See http://www.landfield.com/rfcs/rfc2267.html for further information. This would not stop Denial of Service attacks from happening, but it would aid forensics in back-tracking packets to their source and stopping an attack. This reconfiguration would allow the attack itself to serve as an early warning system that a subnet is being used to launch a Denial of Service attack. Each of the attack methods would cause a large amount of traffic back toward the attack agent. This traffic should set off alarms on intrusion detection software and either start the tracking of the packets, or drop the connection to the router so that no more traffic can cause harm to the target

site. Cisco has written a white paper on how the DDoS attacks can be prevented and how to gather forensic information by modification to the routers on each subnet. It can be found at http://www.cisco.com/warp/public/707/newsflash.html#prevention.

Finally, in the event that your network is taking part in a DDoS against another site, disconnect the systems acting as agents from the network. If the agents cannot be discovered quickly, then it might be necessary to disconnect the router to the outside net. This will stop any more packets from disrupting service to the target and the agents will continue to produce traffic to trace. Remember that the hacker has nearly full control of the system with the agent. Backing up the entire system should be the first step to aid in the forensic process of tracking down the attacker and helping discover the location of the handlers.

Appendix A

Trin00 Described

Information provided from source code version 1.07d2+f3+c.

The agent programs can be installed on Linux and Solaris systems. On some systems, the method used to install the Trin00 agent employs a crontab entry to restart the service every minute. This may be due to a bug in the binary, but it also allows the program to restart if the local system terminates it.

Handler servers are installed on systems that would normally have high packet traffic and large numbers of TCP and UDP connections such as primary name servers, in an attempt to hide the Trin00 servers activities. The compromised systems usually have "rootkit" installed. The rootkit tools hide the presence of the programs, files, and network connections being used. The handler program maintains a list of agent hosts it can contact, and this list is contained in the hidden file "…" (three dots).

Below is a table of how the various players communicate with each other.

Source	Destination	Port
Attacker	Handler(s)	27665/TCP
Handler	Agent(s)	27444/UDP
Agent	Handler(s)	31335/UDP

The program designers require a password for a connection to be made. If another connection is made to the server while someone is already connected to the handler, a warning is sent to the first connection with the second's IP address For example, if the local system administrator or another hacker attempts a connection to the same handler while the original attacker is still connected, an alert will be sent to the original attacker. In this version, the alert passes an incorrect IP address. This still allows the attacker to erase the tracks that might lead to their discovery.

Handler commands:

Die	Shut down the master.
Quit	Log off the master.
mtimer N	Set DoS timer to N seconds. N can be between 1 and 1999 seconds. If N is < 1, it defaults to 300. If N > 2000, it defaults to 500.
dos IP	DoS the IP address specified. A command ("aaa l44adsl IP") is sent to each Bcast host (i.e., Trinoo daemons) telling them to DoS the specified IP address.
mdie pass	Disable all Bcast hosts, if the correct password is specified. A command is sent ("d1e l44adsl") to each host telling them to shut down. A separate password is required for this command.
mping	Send a PING command ("png l44adsl") to every active host.
mdos <ip1:ip2:ip3>	Multiple DoS. Sends a multiple DoS command ("xyz l44adsl 123:ip1:ip2:ip3") to each Bcast host.
info	Print version and compile information.
msize	Set the buffer size for packets sent during DoS attacks.
nslookup host	Do a name service lookup of the specified host from perspective of the host on which the master server is running.
killdead	Attempts to weed out all dead Bcast hosts by first sending all known Bcast hosts a command ("shi l44adsl") that causes any active daemons to reply with the initial

	"*HELLO*" string, then renames the Bcast file (with extension "-b") so it will be re-initialized when the "*HELLO*" packets are received.
usebackup	Switch to the backup Bcast file created by the "killdead" command.
bcast	List all active Bcast hosts.
help [cmd]	Give a (partial) list of commands, or a brief description of the command "cmd" if specified.
mstop	Attempts to stop a DoS attack (not implemented, but listed in the help command).

Agent commands:

aaa pass IP	DoS the specified IP address. Sends UDP packets to random (0-65534) UDP ports on the specified IP addresses for a period of time (default is 120 seconds, or 1 – 1999 seconds as set by the "bbb" command.) The size of the packets is that set by the "rsz" command, or the default size of 1000 bytes.
bbb pass N	Sets time limit (in seconds) for DoS attacks.
shi pass	Sends the string "*HELLO*" to the list of master servers compiled into the program on port 31335/udp.
png pass	Sends the string "PONG" to the master that issued the command on port 31335/udp.
d1e pass	Shut down the Trinoo daemon.
rsz N	Set size of buffer for DoS attacks to N bytes. (The Trinoo daemon simply malloc()s a buffer with this size, then sends the uninitialized contents of the buffer during an attack.)
Xyz pass 123:ip1:ip2:ip3	Multiple DoS. Does the same thing as the "aaa" command, but for multiple IP addresses.

Trin00 attacks a system over random UDP ports. For this reason, it is not feasible to block all UDP traffic. However, one could block the default UDP ports that the handler and clients use to communicate (27444 and 31335).

Detection of the Trin00 tool is difficult. Here are some fingerprints of the agent and the handler. Common names of the Trinoo agent are: ns, http, rpc.Trin00, rpc.listen, trinix, rpc.irix, and irix. Agents can be detected by monitoring crontab files for their repeated startup. Scripts used to automate the installation of the Trinoo network use the “rcp” command.

Handler servers are harder to detect. The only proactive detection method is to search for the hidden file “…” which is the default file name for known hosts the handler can control. This file is located in the same directory as the handler server binary.

When an agent is found, an IP list of the handlers can be found by using the UNIX “strings” command on the agent binary. When a handler is found, the agents can be located using the known hosts list. If the file was encrypted then take control of the handler using the “bcast” command. REMEMBER THAT THIS SENDS AN ALERT TO THE ATTACKER IF THEY ARE LOGGED IN AT THE SAME TIME.

When the Trin00 agent is executed, the agent announces its availability by sending a UDP packet containing the string “*HELLO*” to its programmed Trin00 handler’s IP address. Agents receiving the broadcast respond to the handler with a UDP packet containing the string “PONG”. Monitoring the two UDP communication ports (27444 and 31335) for these strings may produce good results.

Detection of the Trin00 network from within that network is easier once a DoS attack begins. Large numbers of packets containing 4 bytes (all zeros) and coming from one source port to random destination ports on the target host is a good indicator. Look for a number of UDP packets with the same source port and different destination ports. This gives the agents back-track to catch the handlers.

Appendix B

Tribe Flood Network Described

The operation of TFN is similar to that of Trin00. The handlers maintain a list named "iplist" of known agents they can contact. The "iplist" is not encrypted in this version but recent installations of TFN agents have included strings that would indicate the author has added Blowfish encryption to the "iplist".

Control of the handlers is accomplished through command line execution. This can be done by any number of methods including, but not limited to, remote shell bound to the TCP port, SSH terminal sessions, LOKI, and normal telnet sessions. The agent and the handler communicate through ICMP_ECHOREPLY packets. Many network-monitoring tools do not show the data portion of the ICMP packets, so it may be difficult to actually monitor communications between the agent and the handler. TFN can attack with four different protocols: UDP flood, TCP flood with SYN, ICMP flood, and smurf attack. Another "feature" of TFN is that an "on demand" root shell is bound to the TCP Port.

Communication to the client is sent in the form of a 16-bit binary number in the ID field of the ICMP_ECHOREPLY packet. These values are easily changed in the source code, and encouraged. Any arguments are passed as clear ASCII text in the data field of the ICMP_ECHOREPLY packet. This is to prevent someone from stumbling across the agents and taking control.

Tribe Flood Network Commands:

Default value	Description
-2 <bytes>	For replies to the client set packet size for packets used for udp/icmp/smurf attacks.
-1 <mask>	et spoof mask. 0 will use random ips, 1 uses the correct class a, 2 corrects class b, and 3 corrects class c ip value.
0	To change size of udp/icmp packets.
1 <targets>	UDP flood. Target is one ip or multiple ips separated by @.
2 <targets> <port>	SYN flood. If port is 0, random ports are used.
3 <targets>	ICMP echo request flood.
4 <port>	Only if compiled with ID_SHELL. Bind a rootshell to <port>.
5 <target@bcasts>	Smurf amplifier icmp attack. Unlike the above floods, this only supports a single target. Further ips separated by @ will be used as smurf amplifier broadcast addresses.

Communication between the handler and the agents is done with crafted ICMP_ECHOREPLY packets. It would be very difficult to block all ICMP traffic without breaking most Internet programs.

Monitoring for "rcp" connections (514/TCP) from multiple systems on your network, in quick succession, to a single IP address outside your network would be a good trigger.

Intrusion detection software can be set up to look for a large number of ICMP packets with different source IP addresses sent to the same destination IP address.

There is also no authentication of the ICMP packets. If the default values have not changed, then single ICMP_ECHOREPLY packets could be used to flush out the agents

on your network. In the event the codes have been changed, a brute force attack, where you craft ICMP packets to every port on every machine, could produce results, but this would also flood your network with ICMP requests.

Appendix C

Tribe Flood Network 2000 Described

The creator of TFN2K designed this attacker tool to illustrate the point that hacker tools are becoming more sophisticated, being improved in the hacker arena, and kept from the private commercial sector. The designer chose a Denial of Service attack because the results are predictable. The tool was designed to compile on as many operating systems as possible. The source code is portable to Linux, Solaris, most UNIX flavors, and Windows.

TFN2K employs the usual Denial of Service attacks of ICMP Flood, SMURF Flood, SYN Flood, and UDP Flood. In addition to these, Targa3 and mix attacks are also used. Targa3 uses random malformed IP packets that cause some IP stacks to crash or act unexpectedly. The mix attack sends UDP, SYN, and ICMP packets on a 1:1:1 relation. This can have adverse reactions on some routers, network intrusion detection software, and sniffers. Although there is no specific command to remotely update the agents, there is a mechanism in place that would allow a one-way remote shell command to the handlers. This could be used not only to update the agents, but also to fetch password files and generate spam.

All of the attacks utilize spoofing of IP addresses. The communication between the attacker and the handler is done with a randomly chosen protocol (TCP, UDP or ICMP) that is optimized with internal values so that no recognizable pattern can be found in packets. It will pass through most filtering mechanisms. A specific protocol, called Tribe Protocol, is contained in the packet payload. The Tribe Protocol is CAST-256 encrypted and base64 encoded, then decoded by the handlers. The payload consists of the command ID followed by the target or option string. Unlike the other DDoS attack tools, there is no acknowledgement of the communication back to the client. Instead, the client issues the command 20 times relying on the probability that the handler will receive at least one of them.

Decoy packets can be sent out with every real packet when the players are communicating. This completely obscures the attacker/handler communications, making it extremely difficult to determine the true location of the servers.

There are no default passwords with the source code. All passwords are requested at the time of the make. All command strings are a single character, so it would be extremely difficult to locate it in the payload of the communication packets with this alone. The designer then goes the extra step of encrypting the payload. These characteristics make the network scanning tools written by Dave Dittrich and David Brumley ineffective.

Default option commands:

+	/* session header separator, can be anything */
a	/* to bind a root shell */
b	/* to change size of udp/icmp packets */
c	/* to switch spoofing mode */
d	/* to stop flooding */

e	/* to udp flood */
f	/* to syn flood */
g	/* to set port */
h	/* to icmp flood */
i	/* to smurf flood haps! haps! */
j	/* targa3 (ip stack penetration) */
k	/* udp/syn/icmp intervals */
l	/* execute system command */

TFN2K considered NIPC's tool as well. The designer of TFN2K states that there are no strings that can be found in the handler executable, but there are strings that can be found in the agent. The designer goes on to comment that there are public programs that convert binaries to self-extracting compressed executables. This makes the strings undetectable by pattern matching tools.

Fortunately, there is a weakness in what appears to be an oversight in the Base 64 encoding that happens after the encryption. At the end of every TFN2K packet, regardless of the protocol and encryption algorithm, there is a sequence of 0x41s, which translates to A. The actual count of 0x41 appearing at the end of every packet will vary, but there will always be one.

Appendix D

Stacheldraht Described

Information is provided by the source code version 4.0. Be aware that David Drittrich's analysis of Stacheldraht was done with version 1.1. The default ports and passwords have since changed. Stacheldraht is so easy to update, all of the particulars in any paper about Stacheldraht should be taken as just an example. In the three papers found about this attack tool, the ports were rarely the same and there appears to be a set of passwords.

Stacheldraht combines the features of Trin00 and TFN, and adds encrypted communication between the client and the handler, plus automated remote update of the agents. Like the tools before it, Stacheldraht runs on Linux and Solaris.

The topology is the same as TFN and Trin00 with a client controlling the handlers controlling the agents. The attacker uses an encrypted telnet-like session to connect and communicate with the handlers. There is a limit of 6000 agents that a handler can control.

Below is a table of the default ports over which the players communicate.

Source	Destination	Port
Attacker	Handler(s)	65512/TCP
Handler	Agent(s)	65513/TCP
Agent	Handler(s)	65513/TCP

The handler and agent can also communicate over ICMP_ECHOREPLY. The communication between the attacker and the handler is done with symmetric key encryption. Stacheldraht also provides a method to update the agents remotely on demand. This feature employs the Berkley "rcp" command on port 514/TCP. This feature would allow the attacker to continually change the port passwords and command values.

There are many strings that could be used in identifying the binaries in the file system. The NIPC tools would be effective in discovering these agents. However, if the binaries get compressed, this technique will not be successful.

When the agents start up, they attempt to read a handler list configuration file to discover which handler controls it. This file is encrypted with the blowfish algorithm and is a list of IP addresses. If this file cannot be located, there are default handler IP addresses compiled into the binary. Once the agent has the list of potential handlers, it starts sending ICMP_ECHOREPLY packets with the ID field as 666 and the date field containing the string "skillz". This was true in version 1.1 and still is in version 4.0. The handler will respond with "ficken" with a 667 in the ID field. All this communication is done in clear text and can be sniffed.

Commands:

.distro user server	Instructs the agent to install and run a new copy of itself using the Berkeley "rcp" command, on the system "server", using the account "user" (e.g., "rcp user@server:linux.bin ttymon")
.help	Prints a list of supported commands

.killall	Kills all active agents.
madd ip1[:ip2[:ipN]]	Add IP addresses to list of attack victims.
.mdie	Sends die request to all agents.
.mdos	Begins DoS attack.
micmp ip1[:ip2[:ipN]]	Begin ICMP flood attack against specified hosts
.mlist	List IP addresses of hosts being DoS attacked at the moment
.mping	Pings all agents (bcasts) to see if they are alive.
.msadd	Adds a new master server (handler) to the list of available servers.
.msort	Sort out dead/alive agents (bcasts). (Sends pings and shows counts/percentage of dead/alive agents).
mstop ip1[:ip2[:ipN]] mstop all	Stop attacking specific IP addresses, or all.
.msrem	Removes a master server (handler) from the list of available servers.
msyn ip1[:ip2[:ipN]]	Begin SYN flood attack against specified hosts.
mtimer seconds	Set timer for attack duration. (No checks on this value.)
mudp ip1[:ip2[:ipN]]	Begin UDP flood attack against specified hosts. (Trinoo DoS emulation mode.)
.setisize	Sets size of ICMP packets for flooding. (max:1024, default:1024).
.setusize	Sets size of UDP packets for flooding (max:1024, default:1024).
.showalive	Shows all "alive" agents (bcasts).
.showdead	Shows all "dead" agents (bcasts).
.sprange lowport-highport	Sets the range of ports for SYN flooding (defaults to lowport:0, highport:140).

Stacheldraht is far more difficult to detect than either of the first generation attack tools. This tool communicates with ICMP_ECHOREPLY packets between the players for the most part. It would be very difficult to block it without breaking most Internet programs that rely on ICMP. Intrusion detection software could look for signatures in the ICMP_ECHO traffic. This will be increasingly difficult on larger networks because the amount of normal traffic will increase by the size of the network.

The lack of authentication on the source of the ICMP packets to the agent, and the failure to encrypt the strings in the data portion of the ICMP packet, are the only weaknesses of this tool. If the ports, passwords, and command values have not been changed, these agents can be flushed out with the various scanning utilities written by Dave Dittrich and David Brumley.

Appendix E

Flood Attacks Described

ICMP Flood

This Denial of Service attack sends such a large amount of ICMP_ECHOREQUEST packets to the target host that it cannot respond quickly enough to alleviate the amount of traffic on the network. If the attacker does not forge the source IP address, then they too will suffer a performance reduction because they are using resources not only to send the packet but also to receive packet responses. Once the attacker forges the source IP address, then the attack system can use all its resources to just send packets, while the target has to use resources to receive and reply to packets. Multiply this by a distributed attack across hundreds of hosts and the target will easily be brought down.

To prevent this form of attack, most intrusion detection software will filter these packets out. It is also possible to set up routers and firewalls to limit ICMP echo request or drop them entirely from entering the subnet.

Smurf Attack

The Smurf attack is a brute-force attack targeted at a feature in the IP specification known as direct broadcast addressing. A Smurf hacker floods your router with Internet Control Message Protocol (ICMP) echo request packets (pings). Since the destination IP address of each packet is the broadcast address of your network, your router will broadcast the ICMP echo request packet to all hosts on the network. If you have numerous hosts, this will create a large amount of ICMP echo request and response traffic.

If a hacker chooses to spoof the source IP address of the ICMP echo request packet, the resulting ICMP traffic will not only clog up your network (the "intermediary" network) but will also congest the network of the spoofed source IP address known as the "victim" network by sending ICMP_ECHOREPLY packets.

To prevent a network from becoming the intermediary, turn off broadcast addressing on all network routers that allow it (unless needed for multicast features), or configure a firewall to filter the ICMP_ECHOREQUEST. To avoid becoming the victim of a Smurf attack, have an upstream firewall that can either filter ICMP_ECHOREPLYs or limit echo traffic to a small percentage of overall network traffic.

SYN Flood

When a normal TCP connection starts, a destination host receives a SYN (synchronize/start) packet from a source host and sends back a SYN ACK (synchronize acknowledge). The destination host must then hear an ACK (acknowledge) of the SYN ACK before the connection is established. This is referred to as the "TCP three-way handshake."

While waiting for the ACK to the SYN ACK, a connection queue of finite size on the destination host keeps track of connections waiting to be completed. This queue typically empties quickly since the ACK is expected to arrive a few milliseconds after the SYN ACK.

The TCP SYN flood attack exploits this design by having an attacking source host generate TCP SYN packets with random source addresses toward a victim host. The victim destination host sends a SYN ACK back to the random source address and adds an entry to the connection queue. Since the SYN ACK is destined for an incorrect or non-existent host, the last part of the "three-way handshake" is never completed and the entry remains in the connection queue until a timer expires, typically for about one minute. By generating phony TCP SYN packets from random IP addresses at a rapid rate, it is possible to fill up the connection queue and deny TCP services such as e-mail, file transfer, or WWW to legitimate users.

There is no easy way to trace the originator of the attack because the IP address of the source is forged.

Targa3

This attack sends combinations of uncommon IP packets to hosts to generate attacks. The uncommon IP packets consist of invalid fragmentation, protocol, packet size, header values, options, offsets, tcp segments, and routing flags. This attack was originally launched to Windows systems. Once the TCP stack received the invalid packet, the kernel had to allocate resources to handle the packet. If enough malformed packets were received, the system would crash because of exhausted resources.

UDP Flood

The User Datagram Protocol (UDP) is a connectionless protocol. It does not need to establish a connection to transfer data. Since no connection setup is required before data is transferred, it is difficult to bring a host down by flooding the host with just UDP packets. A UDP Flood Denial of Service attack is created when the agent host sends a packet to a random port on the target machine. When the target host receives a UDP packet, it will determine what is listening on the destination port. If nothing is listening, then it will return an ICMP packet to the forged source IP address notifying that the destination port is unreachable. If enough UDP packets are sent to dead ports on the target host, not only will the target host go down, but computers on the same segment will also be disabled because of the amount of traffic.

References

Axent Technologies - TFN2K - An Analysis by Jason Barlow and Woody Thrower http://www2.axent.com/swat/swat.htm

David Dittrich – The DoS Project's "trinoo" distributed Denial of Service attack tool, http://staff.washington.edu/dittrich/misc/trinoo.analysis

David Dittrich – The "Tribe Flood Network" distributed Denial of Service attack tool, http://staff.washington.edu/dittrich/misc/tfn.analysis

David Dittrich – The "stacheldraht" distributed Denial of Service attack tool, http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

ISS X Force Alert - Denial of Service Attack using the Trin00 and Tribe Flood Network programs, http://xforce.iss.net/alerts/advise40.php3

Results of the Distributed-System Intruder Tools Workshop, Pittsburgh, Pennsylvania USA November 2-4, 1999, http://www.cert.org/reports/dsit_workshop.pdf