Global Information Assurance Certification Paper

SANS Advanced Incident Handling Practical Assignment v1.5c

Option 2 – Document an Exploit, Vulnerability or Malicious Program

Topic: Analysis on DDOS tool Stacheldraht v1.666

Geoffrey Cheng

1. Introduction

Distributed Denial-of-service (DDOS) attack is one of the most dangerous threats that could cause devastating effects on the Internet. Although analysis has been started on DDOS on 1998, people do not realize the devastating effect on Internet until several big organizations and corporations were being hit by the attack of DDOS in July 1999. Since then several DDOS tools are identified and analyzed such as Trinoo, Shaft, blitznet, Tribe Flood Network (TFN), Tribe Flood Network 2000 (TFN2K) and Stacheldraht. All these tools could launch DOS attacks from thousands of compromised host and take down virtually any connection, any network on the Internet by just a few command keystrokes.

This document is intended to investigate and analyze the latest version of Stacheldraht – Stacheldraht 1.666. This version is available for download from http://packetstormsecurity.org/distributed starting on February 2001. Even though there is an alert summery from ISS on Stacheldraht 1.666 at: http://xforce.iss.net/alerts/advise61.php and a detailed analysis on original Stacheldraht by David Dittrich at: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt, I have identified some interesting changes and issues that are not covered in the ISS alert summary. I hope this document can serve the community as a detailed supplementary material on understanding the Stacheldraht 1.666 such that system and network administrator could identify it more accurately and effectively when building their security defensive device like network intrusion detection system.

[In this practical assignment, I set up a machine Linux machine with the hostname "testing", IP 192.168.10.200, running Stacheldraht 1.666 with all the components. It works well even with the loopback interface (127.0.0.1) alone. The victim is 192.168.10.6 which is my desktop. Yet I have to cut the duplicated output from tcpdump for better readability, since loopback interface produces packet information twice]

2. Exploit Details

2.1 Name and Version

The original DDOS tool "Stacheldraht" - a German word means 'Barbed wires' – is released during the middle of 1999. A newer version "1.666" came out in early 2000 which included more features and changes to this DDOS tool. Alternative names are "Stacheldraht 1.666+antigl+yps" and "Stacheldraht 1.666+smurf+yps". Actually "Stacheldraht 1.666+smurf+yps" is a 'special' version within the Stacheldraht 1.666 distributed nowadays. By default, the smurf attack is enabled during compilation already, but it is not even shown on the help menu. This is how the 'special' is defined in this distribution. This exploit is documented as CA-2000-01 by cert organization: http://www.cert.org//advisories/CA-2000-01.html.

2.2 Variants

The Stacheldraht 1.666 is an improved version of Stacheldraht, which is a variant from TFN, one of the earliest DDOS tool distributed in public. Newer variant that comes after Stacheldraht is TFN2K and, possibly TFN3K that is described in a theoretical paper by the hacker with the handle name "mixer".

2.3 Operating System

The Stacheldraht works on most Solaris and Linux system. The original Stacheldraht, written by hacker 'randomizer', was found to be running on most of the Solaris platform because the Linux version is quite broken. This version is fixed and improved by the hacker 'yps' and 'randomizer' so that the Linux version works very well too. In any case, Stacheldraht must rely on some known vulnerabilities such as buffer overflows in rpc.statd, rpc.statd, rpc.cmsd, rpc.ttdbserverd and the notorious wu-ftpd security bugs in Solaris and Linux before they are planted. It must be also running at root level so as to work on raw socket for the covert channel. Until now there is no Microsoft Windows version available for download.

2.4 Protocols/Services

The Stacheldraht takes advantage of the ICMP as the major covert channel. Other than that strong data encryption is used to hide its data from any network sniffer to protect the attacker when issuing commands.

2.5 Brief Description

The Stacheldraht by itself is a malicious program that covers its track within a compromised system and communicates by covert channel and encryption on the network. The attacker could control hundreds or thousands of compromised system via a single command line interface and launch different types of DDOS attack to victim afterward. It combines the features available from Trinoo, TFN and adds some new DDOS attacks while giving strong encryption to protect the attacker.

Since it is a DDOS tool, any network-connected devices in the Internet could be affected.

This includes routers, servers and even firewalls. Once targeted as a victim, the result is so devastating that not only the targeted host(s), but also the upstream Internet Service Provider (ISP) could be severely affected in network performance and availability too.

This in turn causes a bigger effect when the upstream ISP does not have the bandwidth itself to handle the flooding generated to the targeted host(s). I have even encountered a (ridiculous) case that a small ISP complaint to one of her clients (YES, the ISP complaint her customer) being attacked by DDOS that utilized all ISP bandwidth as well. This indicates the important role of ISP in participating incident handling. Communication should be established between the ISP and the client as early as possible; otherwise dispute could be a result when DDOS really happens.

3. Protocol Description

The Stacheldraht is divided into 3 parts – (I will use the same terminology as David Dittrich used, that is from the Distributed Intruder System Workshop Tools Paper http://www.cert.org/reports/dsit_workshop.pdf) the telnetc program "client", mserv program "handler" and td program "agent", that they will be described in detail in section "How exploit works". Since it is a malicious program, it covers its track within the compromised system and communicates via covert channel before doing any harm.

3.1 Covering Tracks in Network – client and handler

Between the client and handler, standard TCP is used. The handler listens at a predefined TCP port (60001 in this 1.666 version). Attacker may change it by simply modifying the Macro MSERVERPORT in mserv.c and MASTERSERVERPORT in telnetc/client.c file. In this case a standard 3-way handshaking TCP connection will be established and observed between client and handler. To issue command from the client, these steps will be done:

1) Authentication is required for connecting from the client to the handler. Attacker is required to input a password. Old Stacheldraht uses "sicken" as the default password and now the password is asked during the compilation of the handler. The password is done by standard Unix crypt() function during the building of the handler. The encrypted password string will be further encrypted by blowfish algorithm with the key "authentication" when sending over the network.
2) Blowfish encryption will be used on this TCP connection. All data portion of the TCP packets will be encrypted by password used in the authentication stage. So far there is no known weakness against blowfish algorithm, making intrusion analyst very hard to recover what had been talked between the client and the handler except knowing the password used in the authentication. Details of blowfish algorithm could be found at http://www.counterpane.com/bfsverlag.html

In this case, the encryption not only protects data confidentiality, but also the integrity such that session hijacking will not work on this TCP connection without proper encryption key or knowing the password used. About TCP session hijacking, people might refer to Chapter 7, Mitnick Attack from the book Network Intrusion Detection: An Analyst's Handbook by Stephen Northcutt.

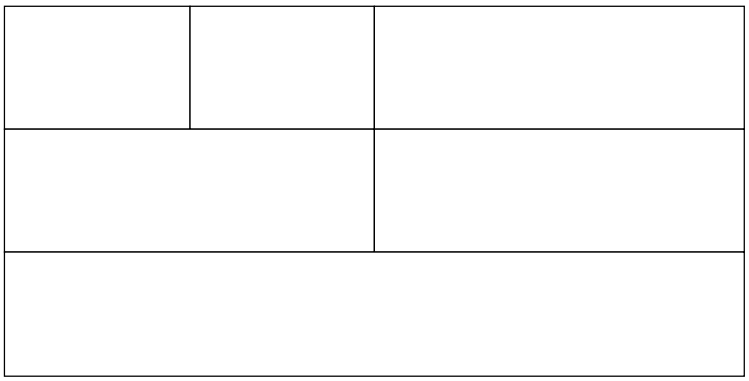
3.2 Covering Tracks in Network – handler and agent

Between the handler and the agent, ICMP and/or TCP are used (In the version 1.666 that I am analyzing, the TCP is disabled).

ICMP originally is used for troubleshooting networking issue and helping UDP on reporting various error situations. Based on the idea covered in the covert channel project Loki, (see Phrack magazine 49-06, http://www.phrack.org), ICMP could be used to carry command and data in its header and optional data portion. Especially there is no state information in the ICMP, if ICMP is allowed this covert channel will exist depending on

the type and code of ICMP. For administrative and network troubleshooting purpose, usually command PING will be allowed using ICMP type-8 code-0, meaning Echo Request and ICMP type-0 code-0, meaning Echo Reply most of the time.

Recalling an ICMP Echo Request (type-8) and Echo Reply (type-0) message structure:



According to the book TCP/IP Illustrated Volume 1 by Richard Steven, page 86, the use of the identifier (ID) field is to set the process ID of the sending process on Unix. This enables the operating system to pass the information to the correct process when there are multiple instances of PING command running. The optional data is left for user's implementation and must be echoed by the receiving client. This makes optional data by default left untouched by any routers in the network path to the destination. Hence, Stacheldraht makes use of the ID field to contain the command and the optional data to contain the parameters such as IP list of the victims.

The following depicts a tcpdump of the communication between the handler (on 127.0.0.1) sending ICMP Echo Reply to the agent (also on 127.0.0.1) to attack a host 192.168.10.200 (c0a80ac8), with command 0x1a0a (6666), meaning massive IP Header attack:

```
18:36:42.090252 > 127.0.0.1 > 127.0.0.1: icmp: echo reply
                          4500 0414 6cef 0000 4001 0bf8 7f00 0001
                          7f00 0001 0000 1a85 1a0a 0000 0000 0000
                          0000 0000 0000 0000 0000 0000 0000 0000
                          0000 0000 c0a8 0ac8 0000 0000 0000 0000
                          0000 0000 0000 0000 0000 0000 0000 0000
                          0000 0000 0000 0000 0000 0000 0000 0000
                          0000 0000 0000 0000 0000 0000 0000 0000
                          0000 0000 0000 0000 0000 0000 0000 0000
                          0000
```

Information is in plaintext (of course in hexadecimal format) from tcpdump since everything between the handler and the agent does not apply any encryption on ICMP packets at all. In old Stacheldraht, encryption will be done on the TCP connection between handler and agents too.

3.3 Covering Tracks in compromised system

This malicious program has to work with other hundreds or thousands of the

compromised in order to generate massive DDOS attack (otherwise it is a standard DOS tool) so it has to cover its track within the system. Stacheldraht 1.666 hides the handler and the agent and their children processes with some common program names when checked by the Unix command "**ps –ef**":

mserv:	(httpd)	# defined as the Macro "moo"
mserv'child:	(httpd)	# defined as the Macro "HIDEKIDS"
td:	lpsched	# defined as the Macro "HIDEME"
td's child:	in.telne	# defined as the Macro "HIDEKIDS"

Furthermore, in order to avoid overloading or crashing the system during attacking, it prevents the attacker to fork more than 10 children on handler and 1 child on agent. This is defined in the Macro CHILDS in various header files:

```
#if CHILDS > 15
#error "Packet kiddie detected..."
#error "That many childs would crash the host... :)"
#endif
```

When the agents start up, it reads a list of handler IP addresses from an encrypted file namely 'mservers' with the encryption key "randomsucks". The handler reads another encrypted file namely 'bcasts' with the password as encryption key, storing a list of agents IP addresses previously registered.

4. Description of variants

Trinoo, TFN and TFN2K are considered to be the variant. Actually parts of the attack code are built from the source code of TFN.

4.1 Overall Structure

Trinoo, TFN and TFN2K are all using 3-tier client/server model. The attacker has to install the front-end client and communicates with the handlers. The handler controls a set of agents on some compromised system to perform DDOS attacks.

4.2 Similarity

With the same overall structure, they could perform massive flooding attack to a victim. The frond-end users issue command, either by program like netcat or tailor-made client within the program structure. They all require root-compromised system before doing any attack. They all support certain covert channel to cover its track between handler and the agents.

4.3 Difference

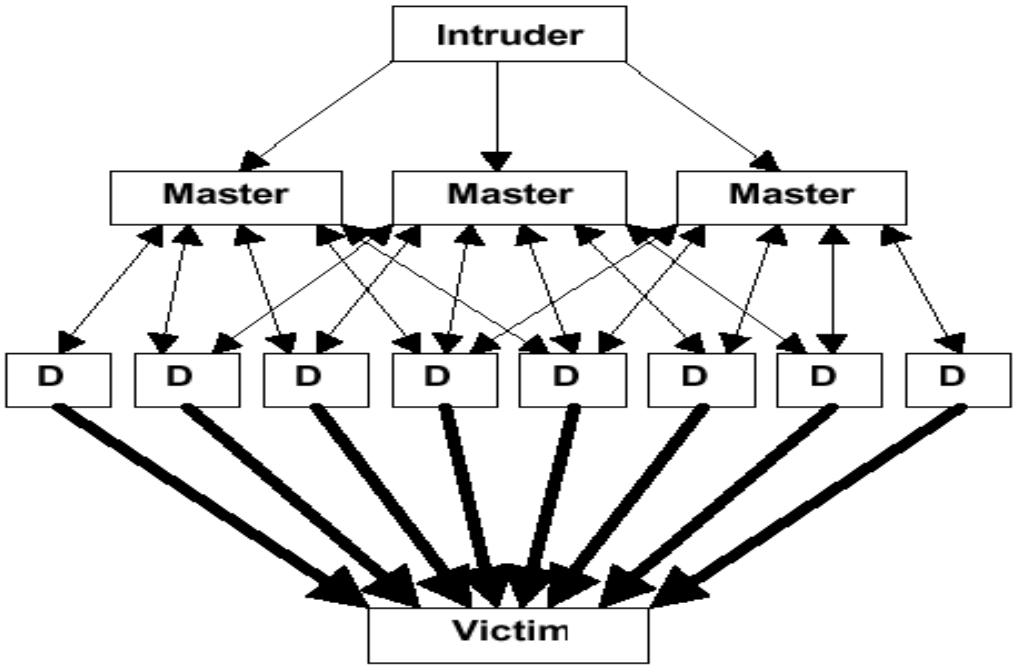
Stacheldraht has data encryption to protect the attacker and between the handler and the agent. Stacheldraht uses ICMP as the major covert channel, while Trinoo uses UDP instead. Original Stacheldraht even supports upgrade or distribution of the agent via Unix command rcp over trusted hosts. TFN and TFN2K are found to be earliest DDOS available on the Windows platform, while Stacheldraht so far it is only working Unix platform discovered in the wild.

5. How the exploit works

5.1 Stacheldraht Network Architecture

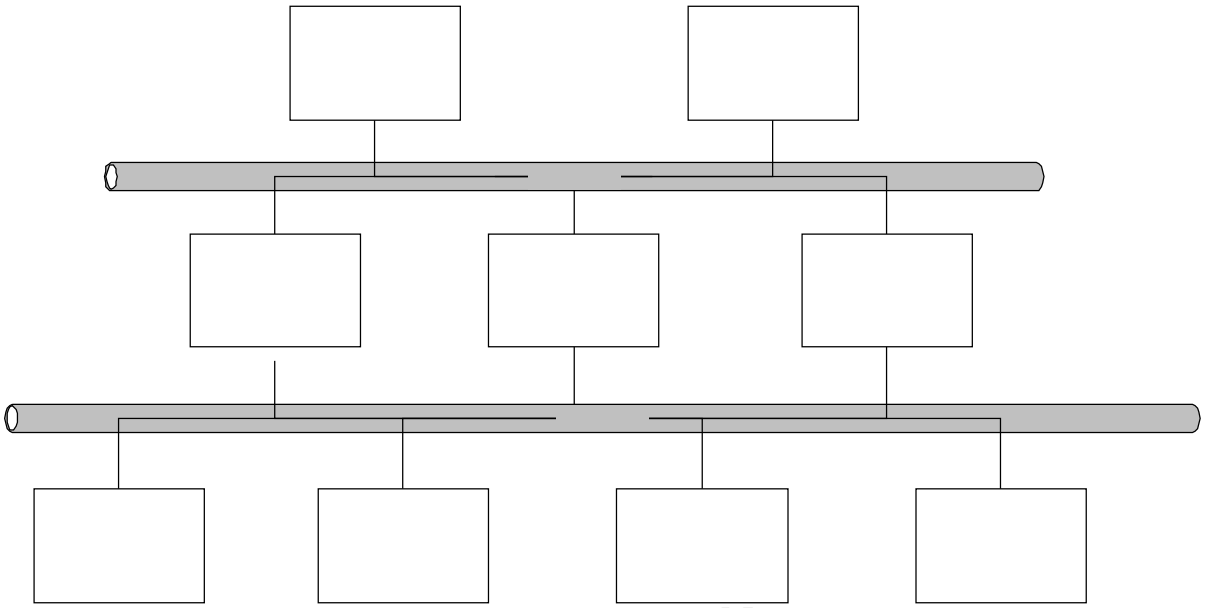
Stacheldraht takes the design from TFN, which is a 3-tier model that commonly used in commercial software architecture design. In commercial software, usually there will be a fancy Graphical User Interface (GUI) communicates with one or more 'Managers' (or Masters or Controllers). Each Manager could control a large set of 'Agents'. This approach could allow the front-end users (the attackers in this case) have a centralized management to thousands of agent while care less on the details between the managers and agents. Moreover, it allows the user to close the front-end system (the telnet client) without affecting the functionality of the back-end systems (handler and agent).

This diagram shows how generic DDOS tools achieve their malicious purpose under this model (diagram is taken from Distributed Intruder System Workshop):



Remark: D – Attacking Device

In Stacheldraht, there is a telnet alike program – telnetc/client, which could talk to the handlers - mserv. Each handler (could be a compromised system too) will control a set of agents – client/td that are compromised system performing the actual attack to targeted hosts. The following diagram describes the architecture of Stacheldraht Network (version 1.666). There could be more than one client and one handler as well:



The Stacheldraht does not exploit new vulnerability in any system therefore it could not replicate itself like Internet worm. Manual distribution is required to install all the handlers and agents into compromised systems.

5.2 DOS/DDOS Features

As a DDOS tool, it features a wide range of flooding attacks:

ICMP flood attack

Send a lot of ICMP packets to victim to utilize its system and network resources. If a firewall system already permits ICMP traffic, this attack might be very effective to overload the victim as well the firewall performing the access control, without knowing any specific services (ports) open on the victim.

SYN flood attack

A classic attack that sends a lot of TCP SYN packets to victim, trying to filling up its backlog and system resources when the victim replies with TCP SYN+ACK packet to establish legitimate connections. Some firewalls (suck as Check Point FireWall-1) might offer some protections (the SYNDefender correspondingly) but in DDOS situation, the buffer given by this protection in handling SYN requests might overload too.

UDP flood attack/UDP flood attack with port 53

Sending a lot of UDP packets to victim to utilize its system and network resources. UDP packets are sent and receive without state information like TCP, letting them passing through most of the routers access-list at certain port. On firewalls, usually DNS replies are allowed which are UDP packets with port 53.

TCP ACK flood attack (in 1.666)

Sending a lot of TCP ACK packets to victim to utilize its system and network

resources. TCP ACK could pass through Cisco router access-list with the keyword established. Depending on the OS, an open port or closed port might reply a TCP RESET packet, causing more traffics and workload on the victim and victim's network.

NULL flood attack (in 1.666)
A TCP flooding attacks with TCP packet's TCP flag all set to 0. This is where the 'NULL' means. The victim might ignore it, consume system resource or crash completely depending on the operating system implementation.

STREAM flood attack (in 1.666)
STREAM or MSTREAM flood attack is borrowed from another DDOS tool 'mstream' discovered. Basically this is TCP ACK flood attack with spoofed source IP, random sequence number and random port number in the packet. Details of the mstream attack could be found at ftp://ftp.technotronic.com/denial/stream-DoS.txt.

HAVOC flood attack (in 1.666)
This is a mixed attack which sending ICMP, UDP, IP, TCP (with random flags) simultaneously to the victim, drawing much CPU utilization on the victim.

IP header attack (in 1.666)
IP header attack is an attack that floods the victim with IP packet with semi-regular IP header with type-of-service (TOS) set to 7 (it is meaningful only in OSPF encoding as stated in RFC 1349.) and so forth. At least it is regular in the sense that the attacking packet will reach the victim.

TCP random header attack (in 1.666)
An attack that borrows from the code "Bubonic.c", which is known to be causing high CPU utilization or 'frozen state' on Windows and some Unix platform. This attack generates lot of TCP packets with randomized setting including the IP offset, TCP sequence number, TCP flag, source port, destination port and so forth. A detail discussion about this exploit could be found at http://www.securityfocus.com/archive/82/78928.

SMURF attack (in 1.666)
A classic attack that spoofing the source IP as the victim one, sending a lot of ICMP Echo Request packets to a broadcast addresses so all clients in the receiving broadcast domain will reply with ICMP Echo Reply packets to the victim.

Some features available on Stacheldraht are now removed in version 1.666, such as the remote update of the Stacheldraht, the testing from handler to agent and the killing of agents. The reason is stated by the author in the code as "removed due to insecure". Insecure it means it could reveal the attacker's location or allow people to identify Stacheldraht agents easily to stop the attack. This is noted that malicious tool just does not do harm, smart hacker could invent tools with security model in mind and improve

them over version.

5.3 The exploit effect

As you might be aware, some attacks alone could cause problematic situation to some specific system already. When there are enough agents, launching attacks to a victim will generally have the following DDOS effects:

- Bandwidth starvation on victim's network
- Utilized the system resources such as TCP backlog
- High CPU loading or frozen state
- Affecting the upstream ISP network performance
- System crash deal to known vulnerability

Big organizations such as yahoo, ebay was reported to be no service available from hours to days when DDOS hit them.

Therefore, I started a simple ICMP attack on an IP 192.168.10.6 (the desktop I am working on Microsoft Word :) from one agent:

```
stacheldraht(status: a!0 d!1)>.micmp 192.168.10.6
mass icmp bombing
1 floodrequests were sent to 1 bcasts.
```

From the tcpdump I immediate see from agent it generates much traffic already:

```
[root@testing client]#tcpdump -w attack1 'icmp'
<Control-C after around 8 seconds…..>
[root@testing client]#tcpdump -r attack1 'dst 192.168.10.6'
15:22:19.078713 eth0 > 127.0.0.38 > 192.168.10.6: icmp: echo request
15:22:19.078847 eth0 > 127.0.0.221 > 192.168.10.6: icmp: echo request
15:22:19.078924 eth0 > 127.0.0.217 > 192.168.10.6: icmp: echo request
15:22:19.079087 eth0 > 127.0.0.178 > 192.168.10.6: icmp: echo request
15:22:19.079163 eth0 > 127.0.0.9 > 192.168.10.6: icmp: echo request
15:22:19.079232 eth0 > 127.0.0.182 > 192.168.10.6: icmp: echo request
15:22:19.079300 eth0 > 127.0.0.35 > 192.168.10.6: icmp: echo request
15:22:19.079369 eth0 > 127.0.0.218 > 192.168.10.6: icmp: echo request
                    [many many entries are deleted]
15:22:27.084546 eth0 > 127.0.0.132 > 192.168.10.6: icmp: echo request
15:22:27.084548 eth0 > 127.0.0.118 > 192.168.10.6: icmp: echo request
15:22:27.084550 eth0 > 127.0.0.88 > 192.168.10.6: icmp: echo request
15:22:27.084552 eth0 > 127.0.0.55 > 192.168.10.6: icmp: echo request
15:22:27.084554 eth0 > 127.0.0.234 > 192.168.10.6: icmp: echo request
15:22:27.091651 eth0 > 127.0.0.145 > 192.168.10.6: icmp: echo request
[root@testing client]# tcpdump -r attack1 'dst 192.168.10.6' | wc
   8960   80640  623263
[root@testing client]#
```

In simply 8 seconds roughly 9000 ICMP packets were sent! The DDOS effect is imaginable if there are thousand of agents available and launch the attack together. If the agent is a faster machine and/or on a faster network, it could generate more packets per second.

5.4 How Agent works and spoof checking

When agents start up, it will check if the current handler is responding, otherwise it will switch to another handler in the list in a cycle. The agent will send an ICMP Echo Reply packet with ID field filled 0x**1a0a** (**666**6, **666** in old version). A string "**skillz**" (0x**736b696c6c7a**) could be found in the payload of the ICMP messages:

```
[root@testing client]# tcpdump -x -i lo
Kernel filter, protocol ALL, datagram packet socket
tcpdump: listening on lo
01:27:40.019260 > testing > testing: icmp: echo reply
                         4500 0414 d52b 0000 4001 a3bb 7f00 0001
                         7f00 0001 0000 9ca3 1a0a 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         736b 696c 6c7a 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
```

If the handler presents with the correct Stacheldraht version, it will send back an ICMP Echo Reply packet with ID field filled with 0x**1a0b** (**6667**, **667** in old version). A string "**ficken**" (0x**6669636b656e**) could be found in the payload of the ICMP messages:

```
01:27:40.049307 > testing > testing: icmp: echo reply
                         4500 0414 d52f 0000 4001 a3b7 7f00 0001
                         7f00 0001 0000 b6b1 1a0b 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         6669 636b 656e 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
```

Before launching attack, when the agent starts up it will check how spoofing works in the compromised environment. It is very intelligence because one of the defensive method to limit the spoofing effect of DDOS is by apply ingress and egress filtering at perimeter routers. One might check out this protection at RFC 2267.

Stacheldraht will set up 2 spoofing level:

0: Spoofing all 32 bits octet of the IP address

3: Spoofing only the last octet of the IP address (that is the host octet in a standard class C address)

The spoofing check is done once every time when the agent first starts up. It begins with sending out an ICMP Echo Request packet with spoofed source **3.3.3.3** to the handler (127.0.0.1) that responded alive. This ICMP Echo Request is crafted with TOS as **7** (an abnormal value) and the IP address of the agent 127.0.0.1 (0x**3132372e302e302e31**) in this ICMP message:

```
03:33:00.997362   lo > 3.3.3.3 > 127.0.0.1: icmp: echo request [tos
0x7,ECT,CE]
                         4507 0400 05c0 0000 ff01 2d2f 0303 0303
                         7f00 0001 0800 f7ff 0000 0000 0000 0000
                         0000 0000 0000 0000 0000 0000 0000 0000
```

```
                        3132 372e 302e 302e 3100 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
```

If the spoofing works, that ICMP Echo Request packet will arrive at the handler (127.0.0.1). The handler then responds with an ICMP Echo Reply packet with ID field set to value of predefined Macro SPOOF_REPLY, which is **9000** (0x**2328**), and a string "**spoofworks**" (0x**73706f6f66776f726b73**) in the ICMP messages:

```
03:33:01.034499   lo > 127.0.0.1 > 127.0.0.1: icmp: echo reply
                        4500 0414 2b81 0000 4001 4d66 7f00 0001
                        7f00 0001 0000 b89a 2328 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        7370 6f6f 6677 6f72 6b73 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
                        0000 0000 0000 0000 0000 0000 0000 0000
```

The agent will then check the ID field if they match. In version 1.666 distributed, it seems that it is bugged that the agent will check the ID field value which is used in old Stacheldraht instead, that is, 1000, not 9000 as shown. This makes this version by default will set the spoofing level to 3 no matter the spoofing check succeeded or not. I believe it is a mistake when the author forgets to amend this value. Please be reminded that it could be changed easily.

[Note: SMURF attack will not work if spoofing level is equal to 3. It is understandable because SMURF attack requires spoofing the 32 bits octet of the victim IP.]

5.5 How the handler works

On the handler, when it starts up it will see if any agents registered to the handler before. It will check a registered agent list by reading a blowfish encrypted file 'bcasts' with the key as the password used in authentication. If there is no agent registered before (that is, no bcasts file), it will prompt the attacker to find some agents to report to this handler first:

```
stacheldraht(status: a!0 d!0)>.mip 192.168.10.200
add some bcasts mofo.
stacheldraht(status: a!0 d!0)>
```

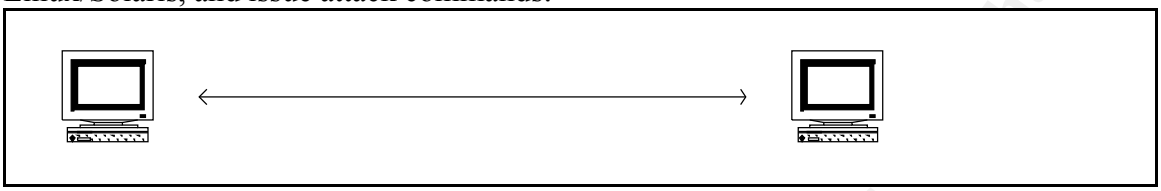
The status a!0 and d!0 means there is 0 alive agent and 0 dead agent.

When there are agents registered, the handler will send the command to agent by ICMP Echo Reply packet, with ICMP identifier (ID) field as the command, and ICMP optional data with the IP lists to attack. In this version, it can control up to 5000 agents. (1000 in original Stacheldraht.)

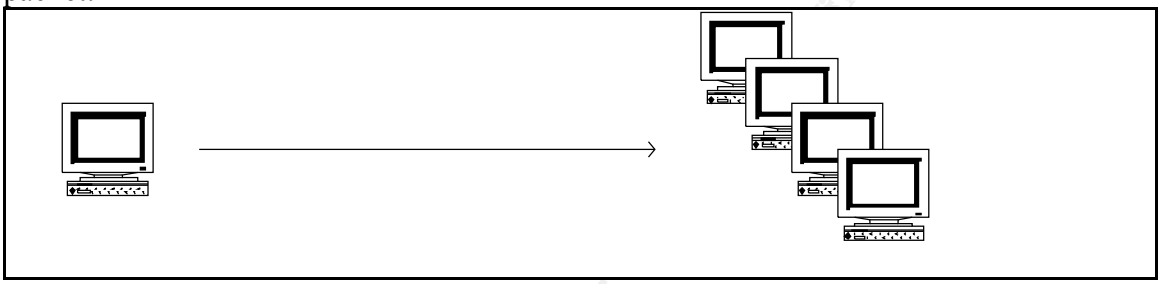
6. Diagram

The following diagram illustrates the typical control and attack stage of Stacheldraht:

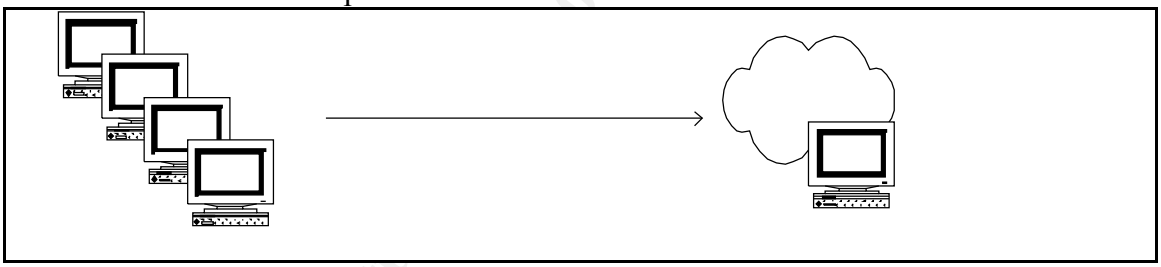
Step 1: Attacker establishes a secure TCP connection to one or more compromised Linux/Solaris, and issue attack commands:



Step 2: The handler broadcasts the command to agents by a one-way ICMP Echo Reply packet:



Step 3: Hundreds or Thousands of agents start flooding victim network until receiving attacker's command to stop:



7. How to use the exploit

The Stacheldraht 1.666 is ready for download from various locations, and it is easy to compile under Linux and Solaris system. Simply extract the stachelantigl.tar.gz first:

```
[root@testing new]# gunzip -dc stachelantigl.tar.gz | tar xf -
[root@testing new]# ls
Makefile  bcasts      blowfish.h  mserv.c               telnetc
README    bf_tab.h    client      setup.c               tubby.h
TODO      blowfish.c  config.h    stachelantigl.tar.gz
[root@testing new]#
```

and compile each component individually.

Compiling mserv:

```
[root@testing new]# make
gcc -lcrypt setup.c -o setup
./setup
-Pre-Compilation----------------------
enter the passphrase :
-------------------------------------
Generated CRYPT-PW: zAudc3X92cBTE
pw.h created..
gcc -lcrypt mserv.c blowfish.c -O6 -o mserv
[root@testing new]#
```

[Note: Different to the original Stacheldrach which has a predefined password 'sicken'.]

Compiling telnetc:

```
[root@testing new]# cd telnetc
[root@testing telnetc]# make
gcc -lcrypt client.c blowfish.c -o client
[root@testing telnetc]#
```

Compiling client (or leaf in original Stacheldraht):

```
[root@testing telnetc]# cd ..
[root@testing new]# cd client
[root@testing client]# make
./setup
-Pre-Compilation----------------------
enter the master host 1 : 127.0.0.1
enter the master host 2 : 0
-------------------------------------
mhosts.h created..
gcc -O6 -fomit-frame-pointer -s -I. td.c blowfish.c -
D__LITTLE_ENDIAN_BITFIELD -
DLINUX -o td -O6
[root@testing client]#
```

[Note: As we can see, the 1.666 client (td) now asks for list of IP of the handlers.]

It is required to add startup script on the compromised system so that the handler (mserv)

and agent (td) program will start again upon reboot. It is reported that when the Stacheldraht was first identified, cron job was running to ensure handler and agent running on a compromised system.

To start the handler, just issue:

```
[root@testing new]# ./mserv
[*]-stacheldraht-[*] - forking in the background...
1 bcasts were successfully read in.
[root@testing new]#
```

To start the agent, just issue:

```
[root@testing client]# ./td
```

To start talking to handler, just supply an IP as the argument and the password like this:

```
[root@testing telnetc]# ./client 127.0.0.1
    [*] stacheldraht [*]
 (c) in 1999 by randomizer

trying to connect...
connection established.
-------------------------------------
enter the passphrase :
-------------------------------------
entering interactive session.
*****************************
   welcome to stacheldraht
*****************************
type .help if you are lame

stacheldraht(status: a!1 d!0)>
```

Let's see, there are lots of command supported in this version, typing .help could help (cruel words are expected in malicious code today):

```
stacheldraht(status: a!1 d!0)>.help
available commands in this version are:
-----------------------------------------------
.mtimer    .mudp   .micmp   .msyn  .mack  .mnul  .msort
.mstream   .mhavoc .mrandom .mip   .mfdns
.showalive .madd   .mlist   .msadd .msrem .help
.setusize .setisize .mdie  .sprange .mstop  .killall
.showdead .forceit  .left
-----------------------------------------------
stacheldraht(status: a!1 d!0)>
```

In short, the following table summarizes the command usages:

.mtimer	Setting attack duration in second.
.mudp	UDP flood attack.
.micmp	ICMP flood attack.
.msyn	TCP SYN flood attack.
.mack	TCP ACK flood attack (in 1.666).
.mnul	NULL flood attack (in 1.666).
.msort	Sort out dead agents and removed them from the lists in the file bcasts.

.mstream	STREAM/MSTREAM flood attack (in 1.666).
.mhavor	HAVOC flood attack (in 1.666).
.mrandom	TCP random header attack (in 1.666).
.mip	IP header attack (in 1.666).
.mfdns	Documented as setting source port of all attacks as port 53, which like DNS named reply. However, it is not actually implemented in the source code I received. I think it should be the .mudns command indeed.
.showalive	Show live agents and their IP addresses
.madd	Add more victims into current attack lists.
.mlist	List current victims' IP.
.msadd	Add handler's IP into agents.
.msrem	Remove a handler's IP from agents.
.help	Print a help menu.
.setusize	Set the packet size of UDP packet, default is 1024 while maximum value is 1024.
.setisize	Set the packet size of ICMP packet, default is 1024 while maximum value is 1024.
.mdie	Originally used to stop all agents, now it is removed to prevent intrusion analyst to identify the present of agents and stop it.
.sprange	Set the low and high port for TCP SYN flooding. This version has removed the range limit (0-140). This request is sent to all agents separated from other attack commands and saved in agents' memory.
.mstop	Stop attack on selected victim. Taking parameter "all" means stopping attack on all victims.
.killall	Kill all child processes of the handler (It is not killing agent as described in other paper)
.showdead	Show dead agents.
.forceit	If it is turned on, it allows you to do .mstop no matter agents are flooding or not. This is useful when the handler and agents are outsync in status.
.left	Show you how much time is left on flooding, useful if you have set up the duration by .mtime.

And some commands that do not show up in help menu:

.distro	Use rcp on the agent to copy to another machine, or download a new version to the agent, it is removed in version 1.666 due to insecure design. It is because it might reveal the attacker's location (he has to put the latest version somewhere, with user and address as arguement), or removing the agent without successfully upgrade.
.msmurf	SMURF attack (in 1.666).
.mudns	UDP flood attack with source port 53 (in 1.666).
.mping	Ping all agents to see if they are alive.
.die	Same as .mdie. It is removed.
.mdos	Change to Trinoo style and immediate starts UDP flood attack. It is equivalent to the command .mudp.

8. Signature of the attack

8.1 Network-based Signature

The encryption provided gives a hard time to intrusion analyst to identify any signature between the client and the handler. The good signature in this situation is the TCP communication with a high port, which will be 60001 in version 1.666.

Between the handler and agents, traffics are not encrypted so it gives better signature. People building intrusion detection might look at ICMP Echo Reply packets with ID field set up with some predefined value, besides 6666 and 6667 shown before: (from config.h)

```
#define  ID_SETPRANGE 9007 /* set port range for synflood */
#define   ID_SETUSIZE 9006 /* set udp size */
#define   ID_SETISIZE 9005 /* set icmp size */
#define    ID_TIMESET 9004 /* set the flood time */
#define     ID_DIEREQ 9003 /* die lame floodserver */
#define   ID_DISTROIT 9002 /* distro request of the master server */
#define ID_REMMSERVER 9001 /* remove added masterserver */
#define ID_ADDMSERVER 9000 /* add new masterserver  */
#define ID_STEST  9099       /* spoof test request by the floodserver
*/
#define ID_TEST  6668       /* test of the master server */
#define ID_ICMP  9055      /* to icmp flood */
#define ID_SENDUDP 9012    /* to udp flood */
#define ID_SENDSYN 9013       /* to syn flood */
#define ID_SENDACK 9113       /* to ack flood */
#define ID_SENDNUL 9213       /* to nul flood */
#define ID_SYNPORT 9014    /* to set port */
#define ID_STOPIT  9015       /* to stop flooding */
#define ID_SWITCH  9016       /* to switch spoofing mode */
#define ID_ACK     9017    /* for replies to the client */
#define ID_SENDSMURF 9028 /* mass smurf request */
#define ID_SENDSTREAM 7778
#define ID_IP 6666
#define ID_SENDHAVOC  9934
#define ID_RANDOM 9935
#define ID_DNS 9936
```

Lone packs of ICMP Echo Reply packet are good signature of compromised agent on the network. It is possible to look at the ICMP payload for strings like ‘skillz’, ‘ficken’, and ‘spoofworks’ that indicates the sign of agent communicating with handlers or performing the spoofing level checks. In any case, non-empty ICMP payload is a must to look at because it is how ICMP covert channel works. Besides, the present of spoofed source address 3.3.3.3 shows the sign of the present of agent too.

It is also possible to check if any ICMP Echo Request with type-of-service field set to 7 in the IP header. Furthermore the sequence number of the ICMP Echo Request and Echo Reply does not change over time and stay at value 0. Normal PING ICMP packets will increase it sequentially for the same instance.

8.2 Host-based signature

Within a system the signature is "program that should not behave like that". Since handler and agent uses ICMP, programs that are listening at the raw socket becomes a good signature:

```
[root@testing client]# lsof |grep raw
mserv     2343  root    3u   raw           14987 00000000:0001-
>00000000:0000 st=07
td        2605  root    0u   raw           15254 00000000:0001-
>00000000:0000 st=07
td        2605  root    3u   raw           15249 00000000:0001-
>00000000:0000 st=07
[root@testing client]#
```

Further check with raw socket listening process review more information on Stacheldraht:

```
[root@testing client]# lsof -p 2343
COMMAND  PID USER   FD   TYPE DEVICE    SIZE  NODE NAME
mserv   2343 root  cwd    DIR    3,8    1024  2038 /root/new
mserv   2343 root  rtd    DIR    3,8    1024     2 /
mserv   2343 root  txt    REG    3,8   59012  2053 /root/new/mserv
mserv   2343 root  mem    REG    3,8  340663 34138 /lib/ld-2.1.3.so
mserv   2343 root  mem    REG    3,8   64478 34147 /lib/libcrypt-
2.1.3.so
mserv   2343 root  mem    REG    3,8 4101324 34145 /lib/libc-2.1.3.so
mserv   2343 root    0u   CHR  136,3          5 /dev/pts/3
mserv   2343 root    1u   CHR  136,3          5 /dev/pts/3
mserv   2343 root    2u   CHR  136,3          5 /dev/pts/3
mserv   2343 root    3u   raw              14987 00000000:0001-
>00000000:0000 st=07
[root@testing client]# lsof -p 2605
COMMAND  PID USER   FD   TYPE DEVICE    SIZE  NODE NAME
td      2605 root  cwd    DIR    3,8    1024  8097 /root/new/client
td      2605 root  rtd    DIR    3,8    1024     2 /
td      2605 root  txt    REG    3,8   93924  8121
/root/new/client/td
td      2605 root  mem    REG    3,8  340663 34138 /lib/ld-2.1.3.so
td      2605 root  mem    REG    3,8 4101324 34145 /lib/libc-2.1.3.so
td      2605 root  mem    REG    3,8  246652 34176 /lib/libnss_files-
2.1.3.so
td      2605 root    3u   raw              15249 00000000:0001-
>00000000:0000 st=07
[root@testing client]#
```

Normally a standard build Unix system will have mostly the login/sshd/sendmail program using **libcrypt** (but libc are common library) running in the background. Using lsof and grep program indicates that libcrypt could be a sign of Stacheldraht handler program too.

Binary of Stacheldraht could be possibly identified by using Unix command strings and signature are keywords like "**3.3.3.3**", "**mserve**r", "**sicken**", "**skillz**", "**randomsucks**" and so forth. All the keywords discussed before apply here.

9. How to protect against it

Protecting against Stacheldraht is like protecting against most of the DDOS tools. If there is no agent infected, the DDOS attack cannot start. Therefore the first goal is to protect system from being infected. System administrator must make sure their Unix systems are up-to-date to prevent being compromised. Moreover, it will be safe to turn off any unnecessary services if it is not needed, such as those RPC services. Bear in mind that this must be put into an exceptional list. Otherwise when the service turns on again it might be vulnerable to some known attacks.

The best approach is to deploy a hardening procedure for every new server installed – as an important procedure in preparation of incident handling. There are several OS hardening tools available such as YASSP http://www.yassp.org/ for Solaris. They can also closely monitor the system process by the command '**top**' since top could review the true name of the program, especially if the agent is planted and executing attacks:

```
10:59pm  up  8:58,  4 users,  load average: 0.53, 0.14, 0.04
52 processes: 48 sleeping, 4 running, 0 zombie, 0 stopped
CPU states:  0.4% user,  0.3% system,  0.0% nice, 99.2% idle
Mem:   192736K av,  189380K used,    3356K free,    5972K shrd,
58620K buff
Swap:  265032K av,    6540K used,  258492K free
59716K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME
COMMAND
 2305 root      18   0   360  316   276 R       0 99.5  0.1   0:41 td
 2306 root       1   0   844  844   652 R       0  0.9  0.4   0:00
top
    1 root       0   0   108   52    36 S       0  0.0  0.0   0:06
init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00
kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:05
kupdate
    4 root       0   0     0    0     0 SW      0  0.0  0.0   0:00
kpiod
    5 root       0   0     0    0     0 SW      0  0.0  0.0   0:00
kswapd
    6 root     -20 -20     0    0     0 SW<     0  0.0  0.0   0:00
mdrecoveryd
  318 bin        0   0    88    0     0 SW      0  0.0  0.0   0:00
portmap
  333 root       0   0     0    0     0 SW      0  0.0  0.0   0:00
lockd
  334 root       0   0     0    0     0 SW      0  0.0  0.0   0:00
rpciod
  343 root       0   0    88    0     0 SW      0  0.0  0.0   0:00
rpc.statd
  357 root       0   0    64   56     0 S       0  0.0  0.0   0:00
apmd
```

Of course, using lsof could reveal program's detail, especially the network socket it is listening. But command lsof does not bundle in the Solaris packages. System

administrator must download and install it by herself.

Since the handler talks to the agents via ICMP Echo Reply packet, routers and firewalls could block all ICMP between the infrastructure and the Internet. Only accept ICMP traffic from trusted hosts unless it is necessary to permits ICMP traffic.

Deploying network-based IDS (NIDS) is also helpful to identify possible Stacheldraht traffic. It could be an alert if there are lots of ICMP Echo Reply packets without the corresponding ICMP Echo Request packet. Commercial NDIS could recognize Stacheldraht and other DDOS tools' signatures. Today host-based IDS reveals its important role on the infrastructure. Host-based IDS should be used on servers and checks for unexpected activities. Even without host-based IDS, system administrator could check **root email**, **crontab** and **atjob** queue to see if there are Stacheldraht start up script added too.

Using ingress and egress filtering at perimeter routers could help in tracing the origin of the agent's attack. However, it does not stop DDOS. By ingress and egress at least you can stop 32 bits octet IP spoofing from working. Moreover, it is a must to deny IP directed-broadcast so that the classic SMURF will not work, as it is now bundled in the version 1.666. Simply adding the command "**no ip-directed broadcast**" on a Cisco router interface will do so.

The **find_ddos** utility could be able to find Stacheldraht (and other DDOS tools) within a system. Bear in mind that find_ddos is distributed in binary format, it is possible it is infected with other Trojan program. One must use it with care. It is available at ftp://ftp.technotronic.com/denial. Using this program it will generates a LOG file and a "files" directories containing all the details of the DDOS tools identified. Here is the sample output of the LOG file:

```
[root@testing find_ddos_v31_linux]# cat LOG
Log started Thu Aug 26 16:00:07 2001

Scanning running processes:

/proc/798/exe:
 identified as: stacheldraht daemon
 with no symbol table
 with the following differences:
  missing string: Error sending syn packet.
  missing string: nohup ./%s
  missing string: rcp %s@%s:sol.bin %s
  missing string: rm -rf %s
  missing string: sicken
  missing string: ttymon
 IP address found: 3.3.3.3 (spoofed address)
 Grabbing: /proc/798/exe
  to: /root/find_ddos_v31_linux/files/798

/proc/800/exe:
 identified as: stacheldraht master
 with the following differences:
```

```
  extra symbol: albcasts
  missing symbol: atexit
  missing symbol: atoi
  extra symbol: bcastcount_alive
  extra symbol: bcasts_alive
  missing symbol: connect
  missing symbol: environ
  missing symbol: errno
  extra symbol: force
  extra symbol: forceit
  missing symbol: getsockopt
```

[many many entries deleted……]

```
Grabbing: /root/new/mserv
  to: /root/find_ddos_v31_linux/files/mserv

Log finished Thu Aug 26 16:02:04 2001

[root@testing find_ddos_v31_linux]#
```

It is no longer possible to use the perl script "gag" distributed in David Dittrich's paper to identify compromised system with the agent installed, simply because the feature of testing by the handler to agent (ID_TEST) is completely removed.

Capacity planning and redundancy might minimize the effects of DDOS when you are being hit as a victim. However this will be the most costly solution and if the attackers find your redundant sites or his agents could generate enough floods, it won't help too.

ISP and law enforcement could help when DDOS really happens. Of course any sign of attacker using the telnet client or handler should be reported as well. In incident handling plan, ISP and law enforcement should be involved as earliest as possible. Communication is very important if you treat your ISP as your working partner, chance of being an infected agent attacker would be lowered, and most importantly, only cooperation and good coordination could stop DDOS when it happens. This comes the getting popular businesses such as Managed Security Services too.

Last but not least, regular security audit and assessment by independent party could help. In this case malicious insider might not be able to install this tool intentionally. System administrators should also deploy vulnerability-scanning tools such as ISS Internet Scanner or Nessus (http://www.nessus.org) that could identify vulnerabilities on a system. (In nessus Stacheldraht is in the backdoor category). It is because Stacheldraht relies no other system vulnerability to be planted. This means that if a system has the agent or handler installed, it is high possible other malicious tools are installed as well (as long as there is no malicious user). This includes rootkits and various backdoor programs. In this case backup and restore will be very important to bring back the system into a clean state.

10. Source code/Pseudo code

The source code is widely distributed. Both original Stacheldraht and version 1.666 could be found easily. They are written in C language and some changes could allow them portable in all Unix platform. Yet the popular and distributed version is on Linux and Solaris. Here is a summary of the source code found in 1.666:

Telnet alike client (under telnetc directory):

Makefile	Small makefile for compiling the program.
bf_tab.h, blowfish.h, blowfish.c	Header files and code for blowfish encryption.
client.c	The main code for client program that talks to handler.

Handler:

Makefile	Small makefile for compiling the program.
README, TODO	Supporting document, the TODO reveals the future of Stacheldraht version.
bf_tab.h, blowfish.h, blowfish.c	Header files and code for blowfish encryption.
tubby.h	Header files that are modified from TFN, implement the communication between handler and agents.
mserv.c	Handler main code.
config.h	Header file containing most of the command Macro used.
setup.c	Code for asking password and generating pw.h during compilation.
pw.h	Header file containing the SALT which is used for blowfish encryption.
bcasts	Files containing reported in agents.

Agent (under client directory for version 1.666 or leaf directory):

Makefile	Small makefile for compiling the program.
bf_tab.h, blowfish.h, blowfish.c	Header files and code for blowfish encryption.
bcasts_lin.h, bcasts_sol.h	Header files for SMURF attack.
setup.c	Code for asking list of handlers and generates mhosts.h.
mhosts.h	Header files containing handlers IP addresses.
config.h,	Header file containing most of the command Macro used.
config.h.in	Original header file from TFN, not used here but present in the source.
control.h	Header files implementing most of functional module of the agents.
icmp.c, udp.c, stream.h, ip.h, syn.c, b.h	Header files and code implementing the details of different attacks.

tubby.h	Header files that are modified from TFN, implement the communication between handler and agents.
td.c	The agent main programming code.

11. Conclusion

Not only I have learned much from this sophisticated tool, but the fact that hackers in the underground communities do work together. The TFN author 'mixer' has even published a paper on how the DDOS will evolve in the future – quoting from his TFN3K paper – "*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…* ". It is really a sarcastic that when the hackers in the underground united together to improve tools over time, while ISP, corporations and governments are still at early stage of co-operation. The effect of DDOS is not just one single entity could handle and co-operation is a must to solve the program – from the agents infection to the attacker launching the attack from his (or just another compromised) systems. It is good to see some ISP nowadays will keep logs for better audits and forensics analysis. Also, sites such as http://www.incidents.org and http://www.cert.org are now getting more and more recognized for corporation to understand the important of the communication and disclosure of incidents.

I believe, the issue from DDOS will be changed when all organizations realized that it is not a technical problem, but a global incident handling participation indeed. This is a process to be done, as we can see the history of Internet security starts from building of bastion hosts, firewall to intrusion detection system and these now come the next step – policy to incident handling.

12. Additional Information/List of Reference

Here is a list of addition information source that is used in this paper and useful for interested party for discussing DDOS:

- RFC 1349: Type-of-services in IP header http://www.normos.org/ietf/rfc/rfc1349.txt
- RFC 2267: Network Ingress filtering http://www.normos.org/ietf/rfc/rfc2267.txt
- Network Intrusion Detection: An Analyst's Handbook by Stephen Northcutt, published by News Riders
- TCP/IP Illustrated Volume 1 by Richard Stevens, published by Addison Wesley.
- David Dittrich Paper on Original Stacheldraht http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt, I
- Distributed Intruder System Workshop Tools Paper by Cert http://www.cert.org/reports/dsit_workshop.pdf
- Blowfish encryption algorithm by Bruce Schneier http://www.counterpane.com/bfsverlag.html
- Loki the ICMP covert channel Volume 49-06 by route http://www.phrack.org
- Stream/Mstream attack discussion ftp://ftp.technotronic.com/denial/stream-DoS.txt.
- Bubonic exploits (TCP random header attack) discussion http://www.securityfocus.com/archive/82/78928.
- Strategies to protect Distributed Denial-of-service (DDOS) attacks by Cisco http://www.cisco.com/warp/public/707/newsflash.html
- TFN3K, the next generation of DDOS tools by mixer http://packetstormsecurity.org/distributed/tfn3k.txt
- The Stacheldraht v1.666 http://packetstormsecurity.org/distributed/stachelantigl.tar.gz
- The original Stacheldraht ftp://ftp.technotronic.com/denial/stachel.tgz
- Lsof for Solaris http://www.sunfreeware.com/programlistsparc8.html#lsof
- Tools for Solaris Hardening http://www.yassp.org/
- The find_ddos tools ftp://ftp.technotronic.com/denial/find_ddos_v31_linx.tar.Z ftp://ftp.technotronic.com/denial/find_ddos_v31_sparc.tar.Z ftp://ftp.technotronic.com/denial/find_ddos_v31_intel.tar.Z
- Free vulnerability scanner Nessus http://www.nessus.org
- Cert coordination centre http://www.cert.org
- Incidents reporting site http://www.incidents.org
- Materials from the Incident Handling Course (GCIH) by SANS institutions.