Snooper-proof Passwords

Andrew Findlay

Brunel University
Uxbridge
GB
Andrew.Findlay@brunel.ac.uk

ABSTRACT

There has been much concern recently about crackers obtaining usernames and passwords by installing network-snooping software on other people's networks. This paper examines the problem and reviews measures that can be taken to combat this type of attack. A low-cost one-time password scheme is proposed. The work is at a very early stage, and there is still hope of finding an existing public-domain system that will fill the requirements adequately.

1. The problem

There has recently been an alarming increase in reports of intruders monitoring network traffic. The Pittsburgh CERT issued an advisory notice[Cen94] on 3rd February 1994, warning that the systems of some service providers had been compromised and all systems that offer remote access through rlogin, telnet, or FTP are at risk. CERT estimated that usernames and passwords for tens of thousands of systems had been captured, and advised all sites to immediately change the passwords of any accounts that could be accessed from any external network.

This particular attack uses a monitoring program that captures usernames and passwords by inspecting the first few packets of all rlogin, telnet, and FTP sessions. The program has to be run as root on a Unix machine, so the intruders had to break into one system on each network they wished to monitor. This was done using well-known security holes that had not been plugged on all machines on the sites concerned. In some cases the intruders probably captured root passwords with their snooper, which would have made the break-in trivial.

The problem is not new. Conventional passwords have always been vulnerable to this sort of attack and to several others:

Network snooping
: As described above

Trojan Horse programs
: A common attack is to leave a terminal running a program that emulates the normal login sequence. The program collects usernames and passwords. Clever versions might even pass these through to the real login process so that the user does not get suspicious about failed logins.

Shoulder surfing
: Some people steal passwords by looking over the real user's shoulder while they are logging in.

Unscrupulous service providers
: A dishonest service provider (e.g. one running a public terminal room at a conference) could easily deploy a range of snooping and Trojan-horse techniques to gather passwords from everyone using their service.

The risks of re-usable passwords have always existed, but are now much greater than before, partly because it is known that an attack is in progress, and partly because machines capable of network monitoring are now common and cheap enough for individuals to buy.

2. Reducing the risk

2.1. Access control

The simplest way to reduce the risk is to restrict which usernames can be used from off-site. At Brunel we use a modified login program, but there are disadvantages with that approach so a "gatekeeper" process that intercepts and validates incoming calls may be more appropriate. Once the list of "at risk" usernames is under control, it is possible to enforce more frequent password changes for those users. In extreme cases, having a restricted list means that only those users on the list must have their passwords invalidated if a break-in is detected. On a site with several thousand users this is a large benefit!

Login restrictions do not protect against on-site crackers, so other measures are needed.

2.2. Encrypting terminals

If each user carried their own "terminal" around with them, it would be possible to encrypt all communications between them and the services they use. Encryption would defeat the snoopers, and there would be no opportunity for an intruder to install a Trojan horse. If a shared-secret encryption method is used, this method even protects against an intruder's machine posing as the service machine.

2.3. Kerberos

Kerberos[Ste88] uses secret-key encryption to provide secure authentication of users without revealing their passwords on the network. It provides useful protection against snooper attacks when it can be run on the workstation that the user is sitting at. Where the user must communicate with a server machine using telnet or a similar protocol, their password is still at risk. Kerberos would thus be a useful addition to the "encrypting terminal" described above.

It is possible to use Kerberos for "inter-realm" authentication between sites that set up such a system. Unfortunately this mechanism does not scale well in the current implementation, and in any case does not provide protection against Trojan-horse attacks.

It will be shown later that Kerberos can be combined with other systems to remove some of these vulnerabilities.

2.4. X.509

The ISO/CCITT Directory Services standards define a mechanism for distributed authentication.[ISO88] It uses the concept of *certificates* which are "signed" by *certification authorities* using public-key encryption. A certificate contains the user's "public key" and is effectively unforgeable, so the user can prove their identity by presenting the certificate and using it to prove that they know their "secret key".

X.509 can provide similar protection to Kerberos, but has the advantage that it scales up to global use much more easily.

2.5. Avoid re-usable passwords

The best protection against snooper attacks is to make sure that no snooper can ever learn anything that would let them into your system. In practice this means that passwords must not be re-usable. There are several authentication systems based on this idea:

- Key-generator cards that display a new key every minute or two
- Key sequence systems that generate the 'next' password on demand
- Challenge-response systems where the service host issues a challenge and the user must give the correct response

Each of these systems is examined in more detail below.

3. Authentication Model

Figure 1 shows a general model of the authentication process that can be applied to most authentication schemes. The user *(U)* carries an authentication device *(Q)* which might be electronic hardware, a sheet of paper, or just a memorised password. Terminal T might be a workstation, PC, X-terminal, or any other interface equipment. The user wants service from a service host S which in turn will refer to an authentication host P to validate the user's identity. Authentication always depends on using secret information, so there is a Key Authority which is responsible for generating the secrets and passing them securely to the user and the authentication host.

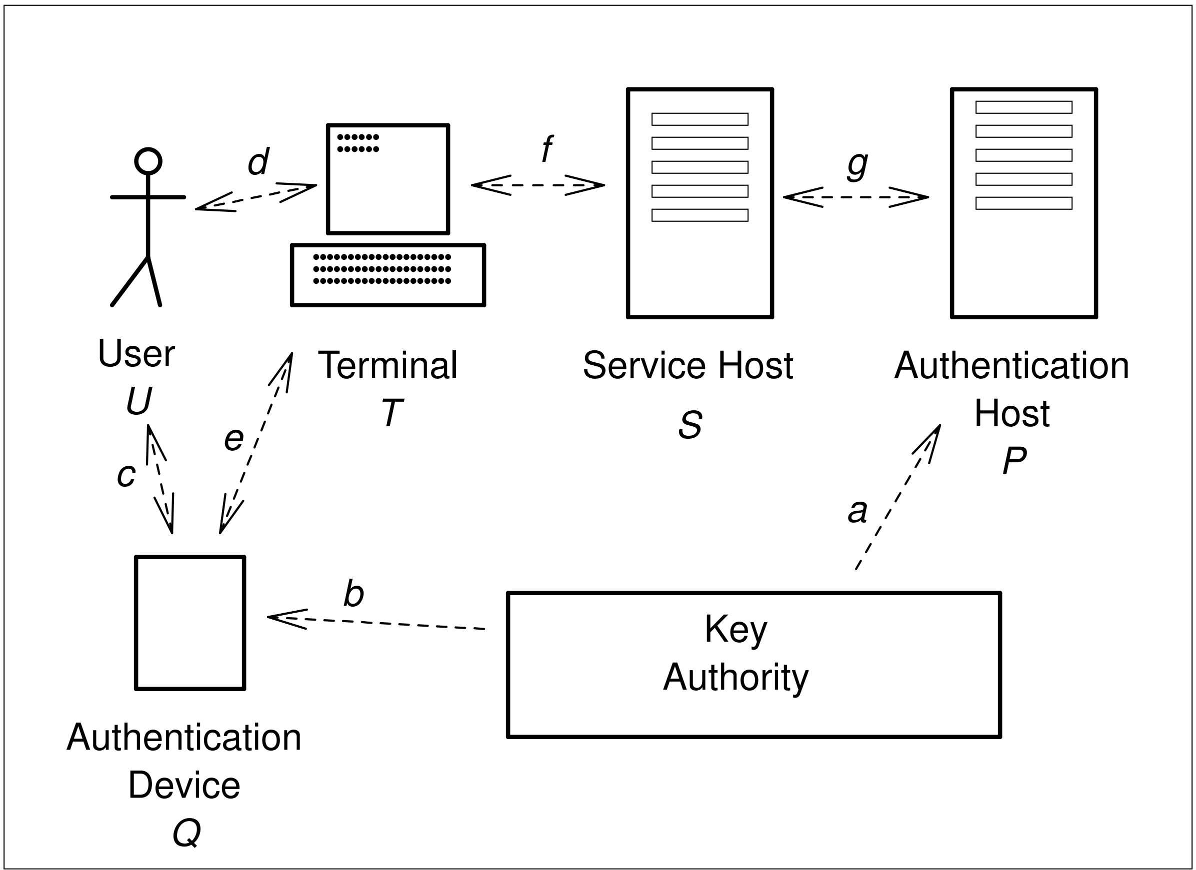

Figure 1: Authentication Model

All communications are subject to snooping, and many are subject to active taps (devices that interpret *and change* the messages that they intercept).

4. Requirements for an authentication scheme

1 The user and the service host must be able to prove their identity to each other without revealing information that could be used to impersonate either one. Similarly, the service host and the authentication host must be confident of each other's identity.

2 No intruder with control over links f or g or terminal T should be able to deny service to the user next time they use a non-compromised channel (e.g. by using up password attempts)

3 No intruder with control over links f or g should be able to hijack a connection that has already been established between U and S

4 No intruder with control over links f or g should be able to read any communication between U and S

Requirements 3 and 4 can only be satisfied by encrypting all communications between U and S which implies special hardware or extra software in all terminal devices. The rest of this paper will concentrate on ways to meet the first two requirements.

5. Analysis of systems

In all the analyses it is assumed that communication b is secured by administrative methods such as requiring the user to collect their password in person and show appropriate identification. Communication a can be secured by encryption or other secure-communication methods.

5.1. Existing username/password system

User U sends a username and password to S in clear. The password is not displayed on screen, to reduce the risk of "shoulder surfing". Communication g consists of the password file entry, and it is protected by having the password field encrypted. The user can change their own password once the first one is issued, thus taking on some of the functions of the key authority.

Strengths: Very simple, easy to use, requires no special hardware, available from all vendors.

Weaknesses: Vulnerable to snooping on links d and f. Vulnerable to Trojan-horse attacks on T, S and P (e.g. by an intruder pretending to be P and supplying bogus password-file entries)

5.2. Time-based key-generator cards

This is a commercial system based on credit-card-size number generating devices. The number generators are time-dependent and they use a hashing algorithm to ensure that is is not possible to derive future numbers by observing the display. A PIN is also used to guard against theft of the card: this can either be entered as part of the login sequence, or (better) on the card itself to modify the number displayed. The cards cost about fifty pounds.

Special software is used to check the number against the set of expected numbers, allowing for some clock drift. Each number is only accepted once to guard against snoopers.

Strengths: Easy to use, good security against snooping and shoulder-surfers.

Weaknesses: expensive for large user-populations, said to be difficult to administer, software and hash algorithms tend to be proprietary so cryptographic strength is unknown. Not completely secure against Trojan horses or active taps (grab the number and immediately use it to open a connection while simulating a network error for the user). Communication g may not be secure in a distributed environment.

5.3. Key Sequence generators

These are similar to the time-based version above, except that they do not depend on absolute time. A generator function is run whenever a key is needed, which produces the "next" key in some sequence. The checking software on host P is prepared for gaps in the sequence of keys actually used in case some of the login attempts fail due to communication problems. The only time-dependent aspect is that no key will ever be accepted if one "later" in the valid sequence has already been accepted. The public-domain *S/Key* system implements this approach using a generator program that the user "carries around" - it can be loaded into portable computers for semi-automatic authentication. S/Key also allows a sheet of keys to be printed out, which is then used as a one-time pad.

Strengths: Good security against snooping, the printout form is cheap.

Weaknesses: Communication g must be secured by other means. Vulnerable to active taps and Trojans (key remains valid until user succeeds in authenticating with a later key)

5.4. Challenge-Response systems

There are a number of these, both public-domain and commercial. Hosts S and P co-operate to issue an unpredictable challenge to the user, who must process it using some secret information and return the correct response. The processing performed by the user is normally an encryption or hash function, so special hardware is usually needed. The encryption device can be protected against theft by a PIN or conventional password. This is the authentication scheme used by the new digital cellular telephones, where the subscriber is identified by an encrypting "smartcard" that they must insert into the phone before it can be used.

A variation on this scheme is to print out a few hundred challenge-response pairs for the user to carry in place of an active authentication device.

The challenge-response system can be extended to prove the identity of host *S* to the user, by returning a key that depends on the original challenge after the response has been verified.

Strengths: Good security against snooping, the printout form is cheap.

Weaknesses: Vulnerable to active taps and Trojans, though if the set of possible challenges is large enough the information is only useful for a few minutes.

6. A paper-based challenge-response system

Where there is a large user population it is important to find a low-cost solution to the authentication problem. The one-time-pad offers a system with low capital cost (although the administrative cost can be high if new pads have to be issued frequently).

6.1. Creating the pads

The key authority is a secure system with a hardware random number generator. It builds *pads* using randomly-generated password strings. Each pad is identified by a serial number and has two parts:

- The user part, which contains the passwords in clear, is printed out and issued to the user by a secure means.
- The system part contains encrypted or hashed versions of the passwords. It is sent to the authentication host by encrypted electronic transfer.

Each pad contains several hundred passwords, which are identified by row and column numbers on the printed sheet. Pads are not associated with a particular user until issued by the authority, so large numbers of them can be stored in host *P* until they are needed. Pads have starting and expiry dates so it is possible to send a new pad to a user before their old one expires. If this must be done by post, the old pad can be used to securely inform host *P* of the safe arrival of the new pad. The hashing algorithm applied to the passwords protects against theft of host *P's* database (perhaps on backup tapes). The user is now in posession of enough secrets shared with host *P* that no secret need ever be used twice.

6.2. Using one-time-pads to authenticate connections

For a single host system, there is little more to be done: the host simply selects a password at random and challenges the user to prove that they know it. If the proof is successful, that password is deleted from the list of available choices.

Where a large network of service hosts is concerned, things get a bit more complicated. There should only be one copy of the pad database to make sure that entries get deleted consistently, so all the service hosts must communicate with a single authentication host (it is quite easy to extend the scheme to allow for backup servers). This communication - *g* in Figure 1 - is extremely vulnerable to any intruder who can masquerade as host *P.* It is therefore necessary to use encryption on link *g.*

The system can be implemented using either public-key cryptography plus secret-key cryptography, or a secret-key system alone. The secret-key system requires each service host to share a secret with the authentication host, so it requires more work to set up but is faster in operation. The public-key version is described here. Suitable encryption algorithms include RSA for the public-key part and DES or IDEA for the secret-key part.

1. The user approaches a terminal *T,* connects to host *S* and gives their username *Un.* Host *S* generates a random "session key" *Ka* and encrypts this plus the username using the well-known public key *Kp* of the authentication service. It sends this cryptogram to host *P.*
2. Host *P* decrypts the message using its private key *Ks* to recover *Ka* and *Un.* If *Un* has a valid pad associated with it, *P* selects an unused entry *E* at random from the pad. If there is no valid pad, a random number is used for *E.* Host *P* forms a challenge *C* by concatenating the serial number of the

pad with the entry number *E*. It encrypts *C* with a secret-key system using key *Ka* and sends the result back to host *S*.

3 Host *S* decrypts the message using *Ka* and applies sanity checks to guard against bogus authentication hosts. A bogus host would not know the key *Ks* and so would not be able to find *Ka.*

4 Host *S* sends the challenge to terminal *T* as clear text, and the terminal displays it for the user.

5 The user reads the pad serial number from the challenge string and selects the appropriate pad if they have more than one. The rest of the challenge string specifies the row and column of the appropriate response password *R,* which the user types in.

6 Host *S* receives the response *R* and encrypts it using *Ka* for transmission to host *P.*

7 Authentication host *P* recovers *R* using *Ka* and applies the hashing algorithm before checking against the hashed password stored in its tables. If the entry matches, it is invalidated and the user's credentials are sent to host *S* encrypted under key *Ka.* If the entry does not match, a failure message is returned to *S.* This failure message is chosen to be the same size as the credentials and is also encrypted to make snooping on link *g* less profitable.

8 Host *S* decrypts the message and either uses the credentials to set up the user's session or sends a "login failed" message back to the user.

6.3. Credentials

In a simple system, the credentials need only contain a pass/fail indication as all the other information required is available in public files.

Where a more secure system such as Kerberos or X.509 is being used to authenticate actions that the user might initiate on host *S* it is necessary to return more information. In the Kerberos case, the authentication host *P* should also be the Kerberos server so that the credentials can include a ticket-granting ticket. For an X.509 system, host *P* would need to hold the user's private key, though this could be encrypted under each one-time password to provide more security at a certain increase in complexity. Other systems such as Sun's ONC+ authentication scheme could be accommodated in similar ways, the aim always being to avoid exposing any information that could be re-used.

6.4. Proving the identity of the service host

If the user needs protection against bogus services (a manipulated share-price information service for example) then it is necessary to include some more secrets in the system. In this case it is necessary for host *S* to share a secret with the authentication host so that host *P* can be sure of host *S's* identity. This secret can conveniently be the encryption key of a secret-key system. An extra field is added to each entry in the pads, and the copy given to host *P* has this field encrypted using the corresponding response password as the key. When *P* has checked the response it uses the password to decrypt this extra "verifier" which it includes with the credentials passed back to host *S.* Host *S* can only decrypt the information because it shares a secret password with *P* so it can only send the correct verifier string back to the user if the user, the service host, and the authentication host are all legitimate.

6.5. How secure is it?

The scheme described protects against passive taps on links *f* and *g* which are the ones most at risk from attacks like the one described by CERT. No information gained by watching those links can be used to gain unauthorised access to the service host.

It protects against simple password-recording Trojans installed on *T* or *S* because response passwords are invalidated after they are used. There is a risk from Trojans that do not pass the response on the the service host, as the pad entry would not be invalidated and might be chosen again later. A pad of limited size would need further protection, perhaps by a mechanism that restricted the number of failed logins that would be tolerated each day.

There is no protection against active taps on link *f* nor against Trojans that simply hijack the connection after it has been validated. These threats can only be countered by the user carrying their own terminal equipment and encryption device.

Exhaustive attempts to break in by trying every possible password are simply frustrated by the number of possible passwords and the fact that each attempt could result in a different challenge.

The sort of password-cracking commonly deployed against Unix password files is ineffective unless a copy of the pad database can be stolen from host *P*.

7. Summary

The risks of re-usable passwords have been analysed, and several better schemes have been examined. A paper-based one-time-pad scheme is proposed, but no implementation has yet been done as further examination of the public-domain authentication systems may yet reveal one with adequate security for large distributed systems.

References

Cen94.CERT Co-ordination Centre, ‘‘CERT Advisory 94:01 - Ongoing Network Monitoring Attacks,’’ *cert@cert.org*, Pittsburgh, 3 February 1994.

ISO88.ISO/CCITT, *The Directory - Authentication Framework*, Geneva, 1988. CCITT Recommendation X.509 / ISO 9594-8

Ste88.J G Steiner, C Neumann, and J I Schiller, ‘‘Kerberos: An Authentication Service for Open Network Systems,’’ *Usenix Conference Proceedings*, Winter 1988.