On Active Attacks to Kerberos Telnet

Simon Josefsson, RSA Laboratories

August 27, 2001

Abstract

We present a well-known and well-documented weakness against active attacks in the Telnet Authentication and Encryption Protocol framework, and discuss the consequences for Kerberos (version 4 and 5) Telnet. We recognize that the weakness can ultimately be used by a active attacker to fool Kerberos Telnet users in some implementations. We briefly describe the protocols involved, the weakness, and demonstrate how it can be used to impersonate a server. We conclude with a recommendation on how to solve the identified problem.

1 Introduction

The Telnet Protocol [1] provides a generic communications facility, primarily used to interface terminal devices with terminal-oriented processes to each other. Implementations of the protocol are often used to perform remote work on Unix-like computers or routers. The core protocol does not support authentication, data encryption nor data integrity. Two framework documents, Telnet Authentication Option [5] and Telnet Data Encryption Option [7], describe how the core protocol may be enhanced with authentication, encryption or data integrity. Several authentication mechanisms have been specified within the framework, but we will focus only on the Kerberos version 4 mechanism [4][1] and the Kerberos version 5 [6] mechanism. Both of these authentication mechanisms also provide keying material for use by the Telnet Encryption option.

The authentication framework specified in [5] describes a three-step authentication process. We will give a brief description here, using a simplified notation of Telnet commands for legibility. Readers familiar with the Telnet protocol and the Authentication Option in particular should note that when we talk about the **WILL, DO, WONT, DONT** authentication command, we are referring to the **IAC * AUTHENTICATE** commands. When we talk about the **SEND, NAME, IS, REPLY** commands, we refer to the **IAC SB AUTHENTICATION * IAC SE** commands. Also, we will talk about these commands as being sent individually, when in practice they are usually bundled together with other commands. The three steps of the authentication protocol are:

[1] The Kerberos 4 mechanism uses a earlier version of the framework [3], but it is similar and compatible with the [5] framework. The latter version add a option of enhancing the encryption negotiation.

- Negotiation of whether authentication is to be used at all. This step uses the **WILL**, **DO**, **WONT**, and **DONT** authentication commands.
- Transfer of server-supported authentication mechanisms. This step uses the **SEND** authentication command.
- Transfer of account name, and mechanism specific data which is used to perform the actual authentication. This step uses the **NAME**, **IS**, and **REPLY** authentication commands.

The client begins by sending the **WILL** authentication command to signal that it wishes to initiate authentication. It expects a **DO** or **DONT** command in return. The **DO** command indicates that authentication is supported by the server and that the client should proceed with authentication. The **DONT** command indicates that the server is not willing to negotiate authentication. If **DO** is not received, the client aborts its attempts of negotiating authentication. These commands are not integrity protected.

In the second step, the server sends the **SEND** command which contain a ordered list of acceptable (to the server) authentication and encryption options. Each entry include indicators for which authentication mechanism to use (e.g. Kerberos 4 or Kerberos 5); if mutual or client-only authentication should be used; and how encryption should be negotiated. Note that the ordered list is not integrity protected, an attacker may remove entries or re-order the list at will. The specification suggest that the ordered list may be integrity protected by the mechanism, however neither the Kerberos 4 or the Kerberos 5 Telnet Authentication Protocols implement this feature.

The clients selects one of the authentication mechanisms supported by the server, and commences step three. In the Kerberos case, the account name is transferred using a **NAME** command. Then a Kerberos structure called **KRB_AP_REQ** is sent in the **IS** authentication command. The **KRB_AP_REQ** structure contains authentication information and keying material. It also contains a check sum calculated on the authentication type chosen by the client. This indicates that some attempts to protect from active attackers were made. However, we will see that leaving the server authentication list unprotected may leave enough room to mount an attack in practice.

The Telnet Encryption Option [7] is similar to the Telnet Authentication Option, the first step negotiate whether encryption should be used at all or not. It uses the **WILL**, **DO**, **WONT**, and **DONT** encryption commands. The client sends a **WILL** encryption command indicating that it is willing to send encrypted data, and a **DO** encryption command indicating it is willing to receive encrypted data. The client expects a **WILL** and **DO** response back from the server, indicating that the server is also willing to send and receive encrypted data. The **WONT** and **DONT** commands are sent when the sender refuses to send or receive encrypted data. Once the parties has agreed to use encryption, the Telnet Encryption framework proceeds similar to the authentication framework, by negotiation which encryption mechanism to use, and then the parameters for that mechanism.

2 Exploiting the weakness

The weaknesses in the negotiation phase, that the list of server supported authentication and encryption mechanisms is not integrity protected, can be exploited in current implementations[2]. We describe the simplest attack, where we are able to disable the Telnet Encryption option – which is responsible for negotiating data encryption and optionally data integrity[3] – and ultimately impersonate a server from the point of view of the end user.

By intercepting the (unprotected) **WILL** and **DO** commands sent by the server, the Telnet client is fooled into a state where it does not proceed to require negotiation of the encryption option even though this was requested by the user. Once the mutual authentication is done, it is possible to cover up for the disabled Encryption option by inserting data into the stream that resembles those messages inserted by the client to inform the user that encryption has been enabled. This step is of no theoretical interest, most users expect that encryption is properly negotiated if she requests it, and that a error or at least a warning is given if encryption isn't enabled. There is a lesson to be learned here though: Never reuse a stream for both insecure data and security information. The user will not be able to tell whether the security information was sent by the possibly insecure remote system (which may be a attacker) or generated by the locally trusted application.

Once any encryption and integrity functionality has been disabled, it is trivial to hijack the session. The following is a excerpt from a terminal session where a user requests a remote session, which is overtaken by a active attacker. The output to the user looks identical to when a mutual authenticated and encrypted channel is opened. In the text below, the session is hijacked right after the mutual authentication, and the active attacker inserts the encryption status information and removes a security warning from the remote system. Familiarity with the Kerberos Telnet system is necessary to appreciate it.

```
alice$ telnet -x bob.example.com
Encryption is verbose
Trying 10.0.0.1...
Connected to bob.example.com.
Escape character is '^]'.
[ Trying mutual KERBEROS5 (host/bob.example.com@EXAMPLE.COM)... ]
[ Kerberos V5 accepts you as ``alice@EXAMPLE.COM'' ]
[ Output is now encrypted with type DES_CFB64 ]
[ Input is now decrypted with type DES_CFB64 ]

eve$
```

Varying this approach, we was able to come up with a similar attack against Ker-

[2]The implementations are KTH Kerberos 4 version 1.0.9 [8] and Heimdal Kerberos 5 version 0.4d [9]. They are used in the FreeBSD, NetBSD and OpenBSD operating systems.

[3]For efficiency reasons, most of the currently defined encryption mechanisms do not provide data integrity.

beros V4 Telnet. Apparently, tricking the client into using client-only authentication, by modifying the Authentication negotiation, disables the attempt to use encryption (even though encryption was requested by the user). However, the Kerberos V4 authentication mechanism actually negotiates a mutually agreed key during the server authentication phase, so the logic of disabling encryption when mutual authentication is unsuccessful is not necessary (and is indeed harmful as it opens up for the attack).

3 Conclusions

The attack we have seen is made possible by combining two things:

- Protocol issue: The phase that negotiate which authentication and encryption scheme should be used is not protected by the mechanism that is eventually chosen.
- Implementation issue: The Kerberos Telnet client should not override user requested security level (i.e. requesting encryption) based on unprotected network data.

This first is well-known and well-documented weakness in the Authentication and Encryption Options. The Authentication option specifications suggests that the entire negotiation phase may be protected by a check sum by the mechanism. Neither the Kerberos V4 or the Kerberos V5 Telnet protocols implement this. This results in a unwanted characteristic in a security negotiation protocol: It is impossible to know whether the eventually selected authentication and encryption suite was the strongest one possible. The reasonable approach in a client, to only accept client-only authentication if mutual authentication was not available, cannot be securely implemented in this protocol. The client is not able to distinguish between the case when no server-side authentication is available and the case when it is under attack. A prudent Kerberos Telnet client has to consider an authentication announcement which lacks mutual authentication as possibly being under attack. Ideally, Kerberos V4 and V5 Telnet Authentication should not leave themselves open to these attacks. However, the Kerberos Telnet specifications does not claim to protects from this attack, so we do not claim to have "broken" it, but we do feel that this is a unwanted characteristic of a security protocol that warrant some concern.

To exemplify to what extent the weakness is known, we quote part of the security considerations for the Telnet Encryption Option [7]:

> `The ENCRYPT option used in isolation provides protection against passive attacks, but not against active attacks. In other words, it will provide protection from someone who is just watching the IP packets as they pass through the network. However, a attacker who is able to modify packets in flight could prevent the ENCRYPT option from being negotiated.`

```
This flaw can be remedied by using the Telnet Authentication option
alongside the ENCRYPT option.  Specifically, setting
ENCRYPT_USING_TELOPT in the authentication-type-pair can be used to
force that Encryption be negotiated even in the face of active
attacks.
```

It is not stated explicitly, but the suggested remedy assumes that the Telnet Authentication option is being integrity protected, otherwise the active attacker is able to modify the Telnet Authentication option as well. This is were the Kerberos Telnet Authentication Option fails, it does protect the Telnet Authentication option from the client to the server, but the initial list of supported mechanisms from the server to the client is never protected. Hence whenever a client and server accepts weaker mechanisms, an attacker is able to force the weaker mechanism to be used. Since the list sent by the server is an ordered list, in practice servers include all the mechanisms they are able to support, thus letting the client (or an attacker) pick the "best" one. This is unlike TLS [10] that protects the entire negotiation phase from any tampering.

Since the attack was based on combining two different ideas, we identify two solutions that prevent the possibility of combining them into a successful attack:

- The negotiation of the authentication option should be protected by a Kerberos check sum. The current specification can be updated as follows: If client-only authentication is chosen, we suggest that a additional command is sent from the server to the client containing a check sum of the initial list of support authentication mechanisms. If mutual authentication is chosen, we suggest that the server **RESPONSE** command should also contain a Kerberos check sum of the initial list. If **ENCRYPT_MASK** of [5] is used, these changes can be used to protect the encryption negotiation as well.
- Correct the implementation to enforce that encryption is successfully negotiated if the user requests encryption, and that the connection attempt is aborted if encryption cannot be negotiated. Also, clients should assume that servers which do not announce mutual authentication may be under attack (unless the protocol is updated as above). If it is acceptable to continue or not in this case should be up to client policy. In many environments it would be better if the Telnet client refuse unencrypted or client-only authenticated connections by default.

Ideally both solutions should be implemented, but either one of them will suffice. The first approach is needed to make the Kerberos authentication protocol more cryptographically sound by itself. We recognize that keeping backwards compatibility may be the deciding factor here.

References

[1] J. Postel and J. Reynolds, TELNET Protocol Specification, RFC 854, May 1983, http://www.ietf.org/rfc/rfc854.txt.

[2] J Kohl and C. Neuman, The Kerberos Network Authentication Service (V5), RFC 1510, September 1993, http://www.ietf.org/rfc/rfc1510.txt.

[3] D. Borman, Telnet Authentication Option, RFC 1416, February 1983, http://www.ietf.org/rfc/rfc1416.txt.

[4] D. Borman, Telnet Authentication: Kerberos Version 4, RFC 1411, January 1983, http://www.ietf.org/rfc/rfc1411.txt.

[5] T. Ts'o and J. Altman, Telnet Authentication Option, RFC 2941, September 2000, http://www.ietf.org/rfc/rfc2941.txt.

[6] T. Ts'o, Telnet Authentication: Kerberos 5, RFC 2942, September 2000, http://www.ietf.org/rfc/rfc2942.txt.

[7] T. Ts'o, Telnet Data Encryption Option, RFC 2946, September 2000, http://www.ietf.org/rfc/rfc2946.txt.

[8] KTH Kerberos IV Implementation, http://www.pdc.kth.se/kth-krb/

[9] Heimdal Kerberos V Implementation, http://www.pdc.kth.se/heimdal/

[10] T. Dierks and C. Allen, The TLS Protocol Version 1.0, RFC 2246, January 1999, http://www.ietf.org/rfc/rfc2246.txt.