Network Working Group
Request for Comments: 1282
Obsoletes: RFC 1258

B. Kantor
Univ. of Calif San Diego

December 1991

BSD Rlogin

Status of this Memo

This memo documents an existing protocol and common implementation that is extensively used on the Internet. This memo provides information for the Internet community. It does not specify an Internet standard. Distribution of this memo is unlimited.

Protocol Description

The rlogin facility provides a remote-echoed, locally flow-controlled virtual terminal with proper flushing of output [1]. It is widely used between Unix hosts because it provides transport of more of the Unix terminal environment semantics than does the Telnet protocol, and because on many Unix hosts it can be configured not to require user entry of passwords when connections originate from trusted hosts.

The rlogin protocol requires the use of the TCP. The contact port is 513. An eight-bit transparent stream is assumed.

Connection Establishment

Upon connection establishment, the client sends four null-terminated strings to the server. The first is an empty string (i.e., it consists solely of a single zero byte), followed by three non-null strings: the client username, the server username, and the terminal type and speed. More explicitly:

```
<null>
client-user-name<null>
server-user-name<null>
terminal-type/speed<null>
```

For example:

```
<null>
bostic<null>
kbostic<null>
vt100/9600<null>
```

The server returns a zero byte to indicate that it has received these

strings and is now in data transfer mode. Window size negotiation may follow this initial exchange (see below).

From Client to Server (and Flow Control)

Initially, the client begins operation in "cooked" (as opposed to to "raw") mode. In this mode, the START and STOP (usually ASCII DC1,DC3) characters are intercepted and interpreted by the client to start and stop output from the remote server to the local terminal, whereas all other characters are transmitted to the remote host as they are received. (But see below for the handling of the local-escape character.)

In "raw" mode, the START and STOP characters are not processed locally, but are sent as any other character to the remote server. The server thus determines the semantics of the START and STOP characters when in "raw" mode; they may be used for flow control or have quite different meanings independent of their ordinary usage on the client.

Screen/Window Size

The remote server indicates to the client that it can accept window size change information by requesting a window size message (as out of band data) just after connection establishment and user identification exchange. The client should reply to this request with the current window size.

If the remote server has indicated that it can accept client window size changes and the size of the client’s window or screen dimensions changes, a 12-byte special sequence is sent to the remote server to indicate the current dimensions of the client’s window, should the user process running on the server care to make use of that information.

The window change control sequence is 12 bytes in length, consisting of a magic cookie (two consecutive bytes of hex FF), followed by two bytes containing lower-case ASCII "s", then 8 bytes containing the 16-bit values for the number of character rows, the number of characters per row, the number of pixels in the X direction, and the number of pixels in the Y direction, in network byte order. Thus:

```
FF FF s s rr cc xp yp
```

Other flags than "ss" may be used in future for other in-band control messages. None are currently defined.

From Server to Client

Data from the remote server is sent to the client as a stream of characters. Normal data is simply sent to the client's display, but may be processed before actual display (tabs expanded, etc.).

The server can imbed single-byte control messages in the data stream by inserting the control byte in the stream of data and pointing the TCP "urgent-data" pointer at the control byte. When a TCP urgent-data pointer is received by the client, data in the TCP stream up to the urgent byte is buffered for possible display after the control byte is handled, and the control byte pointed to is received and interpreted as follows:

02 A control byte of hex 02 causes the client to discard all buffered data received from the server that has not yet been written to the client user's screen.

10 A control byte of hex 10 commands the client to switch to "raw" mode, where the START and STOP characters are no longer handled by the client, but are instead treated as plain data.

20 A control byte of hex 20 commands the client to resume interception and local processing of START and STOP flow control characters.

80 The client responds by sending the current window size as above.

All other values of the urgent-data control byte are ignored. In all cases, the byte pointed to by the urgent data pointer is NOT written to the client user's display.

Connection Closure

When the TCP connection closes in either direction, the client or server process which notices the close should perform an orderly shut-down, restoring terminal modes and notifying the user or processes of the close before it closes the connection in the other direction.

Implementation Notes

The client defines a client-escape character (customarily the tilde, "~"), which is handled specially only if it is the first character to be typed at the beginning of a line. (The beginning of a line is defined to be the first character typed by the client user after a new-line [CR or LF] character, after a line-cancel character, after resumption of a suspended client session, or after initiation of the connection.)

The client-escape character is not transmitted to the server until the character after it has been examined, and if that character is one of the defined client escape sequences, neither the client-escape nor the character following it are sent. Otherwise, both the client-escape character and the character following it are sent to the server as ordinary user input.

If the character following the client-escape character is the dot ".", or the client-defined end-of-file character (usually control-D), the connection is closed. This is normally treated by the server as a disconnection, rather than an orderly logout.

Other characters (client-defined, usually control-Z and control-Y) are used to temporarily suspend the rlogin client when the host has that ability. One character suspends both remote input and output; the other suspends remote input but allows remote output to continue to be directed to the local client’s terminal.

Most client implementations have invocation switches that can defeat normal output processing on the client system, and which can force the client to remain in raw mode despite switching notification from the server.

A Cautionary Tale [2]

The rlogin protocol (as commonly implemented) allows a user to set up a class of trusted users and/or hosts which will be allowed to log on as himself without the entry of a password. While extremely convenient, this represents a weakening of security that has been successfully exploited in previous attacks on the internet. If one wishes to use the password-bypass facilities of the rlogin service, it is essential to realize the compromises that may be possible thereby.

Bypassing password authentication from trusted hosts opens ALL the systems so configured when just one is compromised. Just as using the same password for all systems to which you have access lets a villain in everywhere you have access, allowing passwordless login among all your systems gives a marauder a wide playing field once he has entered any of your systems. One compromise that many feel achieves a workable balance between convenience and security is to allow password bypass from only ONE workstation to the other systems you use, and NOT allow it between those systems. With this measure, you may have reduced exposure to a workable minimum.

The trusted host specification is ordinarily one of a host name. It is possible, by compromise of your organization’s domain name server, or compromise of your network itself, for a villain to make an

untrusted host masquerade as a trusted system. There is little that a user can do about this form of attack. Luckily, so far such attacks have been rare, and often cause enough disruption of a network that attempts are quickly noticed.

When the file containing a user's list of trusted logins is inadvertently left writeable by other users, untrustworthy additions may be made to it.

Secure authentication extensions to the rlogin protocol (Kerberos, et al) can greatly reduce the possibility of compromise whilst still allowing the convenience of bypassing password entry. As these become more widely deployed in the internet community, the hazards of rlogin will decrease.

References

[1] Stevens, W., "UNIX Network Programming", ISBN 0-13-949876-1.

[2] Garfinkel & Spafford, "Practical Unix Security", ISBN 0-937175-72-2.

Security Considerations

See the "A Cautionary Tale" section above.

Author's Address

Brian Kantor
University of California at San Diego
Network Operations C-024
La Jolla, CA 92093-0214

Phone: (619) 534-6865

EMail: brian@UCSD.EDU