Network Working Group
Request for Comments: 1194
Obsoletes: RFC 742

D. Zimmerman
Center for Discrete Mathematics and
Theoretical Computer Science

November 1990

The Finger User Information Protocol

Status of this Memo

This memo defines a protocol for the exchange of user information. This RFC specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This memo describes the Finger User Information Protocol. This is a simple protocol which provides an interface to a remote user information program.

Based on RFC 742, a description of the original Finger protocol, this memo attempts to clarify the expected communication between the two ends of a Finger connection. It also tries not to invalidate the many existing implementations or add unnecessary restrictions to the original protocol definition.

Table of Contents

1. Introduction

1.1. Intent

This memo describes the Finger User Information Protocol. This is a simple protocol which provides an interface to a remote user information program (RUIP).

Based on RFC 742, a description of the original Finger protocol, this memo attempts to clarify the expected communication between the two ends of a Finger connection. It also tries not to invalidate the many current implementations or add unnecessary restrictions to the original protocol definition.

The most prevalent implementations of Finger today seem to be primarily derived from the BSD UNIX work at the University of California, Berkeley. Thus, this memo is based around the BSD version's behavior.

However, the BSD version provides few options to tailor the Finger RUIP for a particular site's security policy, or to protect the user from dangerous data. Furthermore, there are MANY potential security holes that implementors and administrators need to be aware of, particularly since the purpose of this protocol is to return information about a system's users, a sensitive issue at best. Therefore, this memo makes a number of important security comments and recommendations.

1.2. History

The FINGER program at SAIL, written by Les Earnest, was the inspiration for the NAME program on ITS. Earl Killian at MIT and Brian Harvey at SAIL were jointly responsible for implementing the original protocol.

Ken Harrenstien is the author of RFC 742, "Name/Finger", which this memo began life as.

1.3. Requirements

In this document, the words that are used to define the significance of each particular requirement are capitalized. These words are:

* "MUST"

  This word or the adjective "REQUIRED" means that the item is an absolute requirement of the specification.

* "SHOULD"

  This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

* "MAY"

  This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

An implementation is not compliant if it fails to satisfy one or more of the MUST requirements. An implementation that satisfies all the MUST and all the SHOULD requirements is said to be "unconditionally compliant"; one that satisfies all the MUST requirements but not all the SHOULD requirements is said to be "conditionally compliant".

2. Use of the protocol

2.1. Flow of events

Finger is based on the Transmission Control Protocol, using TCP port 79 decimal (117 octal). A TCP connection is opened to a remote host on the Finger port. An RUIP becomes available on the remote end of the connection to process the request. The RUIP is sent a one line

query based upon the Finger query specification. The RUIP processes the query, returns an answer, then closes the connection normally.

2.2. Data format

Any data transferred MUST be in ASCII format, with no parity, and with lines ending in CRLF. This excludes other character formats such as EBCDIC, etc. This also means that any characters between ASCII 128 and ASCII 255 should truly be international data, not USASCII with the parity bit set.

2.3. Query specifications

An RUIP MUST accept the entire Finger query specification.

The Finger query specification is defined:

```
{Q1}    ::= [{U}] [/W] {C}

{Q2}    ::= [{U}]{H} [/W] {C}

{U}     ::= username

{H}     ::= @hostname | @hostname{H}

{C}     ::= <CRLF>
```

{H}, being recursive, means that there is no arbitrary limit on the number of @hostname tokens in the query. In examples of the {Q2} request specification, the number of @hostname tokens is limited to two, simply for brevity.

Be aware that {Q1} and {Q2} do not refer to a user typing "finger user@host" from an operating system prompt. It refers to the line that an RUIP actually receives. So, if a user types "finger user@host<CRLF>", the RUIP on the remote host receives "user<CRLF>", which corresponds to {Q1}.

As with anything in the IP protocol suite, "be liberal in what you accept".

2.4. RUIP {Q2} behavior

A query of {Q2} is a request to forward a query to another RUIP. An RUIP MUST either provide or actively refuse this forwarding service (see section 3.2.1). If an RUIP provides this service, it MUST conform to the following behavior:

Given that:

> Host <H1> opens a Finger connection <F1-2> to an RUIP on host <H2>.
>
> <H1> gives the <H2> RUIP a query <Q1-2> of type {Q2} (e.g., FOO@HOST1@HOST2).

It should be derived that:

> Host <H3> is the right-most host in <Q1-2> (i.e., HOST2)
>
> Query <Q2-3> is the remainder of <Q1-2> after removing the right-most "@hostname" token in the query (i.e., FOO@HOST1)

And so:

> The <H2> RUIP then must itself open a Finger connection <F2-3> to <H3>, using <Q2-3>.
>
> The <H2> RUIP must return any information received from <F2-3> to <H1> via <F1-2>.
>
> The <H2> RUIP must close <F1-2> in normal circumstances only when the <H3> RUIP closes <F2-3>.

2.5. Expected RUIP response

For the most part, the output of an RUIP doesn't follow a strict specification, since it is designed to be read by people instead of programs. It should mainly strive to be informative.

Output of ANY query is subject to the discussion in the security section.

2.5.1. {C} query

A query of {C} is a request for a list of all online users. An RUIP MUST either answer or actively refuse (see section 3.2.2). If it answers, then it MUST provide at least the user's full name. The system administrator SHOULD be allowed to include other useful information (per section 3.2.3), such as:

- terminal location
- office location
- office phone number
- job name
- idle time (number of minutes since last typed input, or

since last job activity).

2.5.2. {U}{C} query

A query of {U}{C} is a request for in-depth status of a specified user {U}. If you really want to refuse this service, you probably don't want to be running Finger in the first place.

An answer MUST include at least the full name of the user. If the user is logged in, at least the same amount of information returned by {C} for that user MUST also be returned by {U}{C}.

Since this is a query for information on a specific user, the system administrator SHOULD be allowed to choose to return additional useful information (per section 3.2.3), such as:

- office location
- office phone number
- home phone number
- status of login (not logged in, logout time, etc)
- user information file

A user information file is a feature wherein a user may leave a short message that will be included in the response to Finger requests. (This is sometimes called a "plan" file.) This is easily implemented by (for example) having the program look for a specially named text file in the user's home directory or some common area; the exact method is left to the implementor. The system administrator SHOULD be allowed to specifically turn this feature on and off. See section 3.2.4 for caveats.

There MAY be a way for the user to run a program in response to a Finger query. If this feature exists, the system administrator SHOULD be allowed to specifically turn it on and off. See section 3.2.5 for caveats.

2.5.3. {U} ambiguity

Allowable "names" in the command line MUST include "user names" or "login names" as defined by the system. If a name is ambiguous, the system administrator SHOULD be allowed to choose whether or not all possible derivations should be returned in some fashion (per section 3.2.6).

2.5.4. /W query token

The token /W in the {Q1} or {Q2} query types SHOULD at best be interpreted at the last RUIP to signify a higher level of verbosity

in the user information output, or at worst be ignored.

2.5.5. Vending machines

Vending machines SHOULD respond to a {C} request with a list of all items currently available for purchase and possible consumption. Vending machines SHOULD respond to a {U}{C} request with a detailed count or list of the particular product or product slot. Vending machines should NEVER NEVER EVER eat requests. Or money.

3. Security

3.1. Implementation security

Sound implementation of Finger is of the utmost importance. Implementations should be tested against various forms of attack. In particular, an RUIP SHOULD protect itself against malformed inputs. Vendors providing Finger with the operating system or network software should subject their implementations to penetration testing.

Finger is one of the avenues for direct penetration, as the Morris worm pointed out quite vividly. Like Telnet, FTP and SMTP, Finger is one of the protocols at the security perimeter of a host. Accordingly, the soundness of the implementation is paramount. The implementation should receive just as much security scrutiny during design, implementation, and testing as Telnet, FTP, or SMTP.

3.2. RUIP security

Warning!! Finger discloses information about users; moreover, such information may be considered sensitive. Security administrators should make explicit decisions about whether to run Finger and what information should be provided in responses. One existing implementation provides the time the user last logged in, the time he last read mail, whether unread mail was waiting for him, and who the most recent unread mail was from! This makes it possible to track conversations in progress and see where someone’s attention was focused. Sites that are information-security conscious should not run Finger without an explicit understanding of how much information it is giving away.

3.2.1. {Q2} refusal

For individual site security concerns, the system administrator SHOULD be given an option to individually turn on or off RUIP processing of {Q2}. If RUIP processing of {Q2} is turned off, the RUIP MUST return a service refusal message of some sort. "Finger forwarding service denied" is adequate. The purpose of this is to

allow individual hosts to choose to not forward Finger requests, but if they do choose to, to do so consistently.

Overall, there are few cases which would warrant processing of {Q2} at all, and they are far outweighed by the number of cases for refusing to process {Q2}. In particular, be aware that if a machine is part of security perimeter (that is, it is a gateway from the outside world to some set of interior machines), then turning {Q2} on provides a path through that security perimeter. Therefore, it is RECOMMENDED that the default of the {Q2} processing option be to refuse processing. It certainly should not be enabled in gateway machines without careful consideration of the security implications.

3.2.2. {C} refusal

For individual site security concerns, the system administrator SHOULD be given an option to individually turn on or off RUIP acceptance of {C}. If RUIP processing of {C} is turned off, the RUIP MUST return a service refusal message of some sort. "Finger online user list denied" is adequate. The purpose of this is to allow individual hosts to choose to not list the users currently online.

3.2.3. Atomic discharge

All implementations of Finger SHOULD allow individual system administrators to tailor what atoms of information are returned to a query. For example:

- Administrator A should be allowed to specifically choose to return office location, office phone number, home phone number, and logged in/logout time.

- Administrator B should be allowed to specifically choose to return only office location, and office phone number.

- Administrator C should be allowed to specifically choose to return the minimum amount of required information, which is the person’s full name.

3.2.4. User information files

Allowing an RUIP to return information out of a user-modifiable file should be seen as equivalent to allowing any information about your system to be freely distributed. That is, it is potentially the same as turning on all specifiable options. This information security breach can be done in a number of ways, some cleverly, others straightforwardly. This should disturb the sleep of system administrators who wish to control the returned information.

3.2.5. Execution of user programs

Allowing an RUIP to run a user program in response to a Finger query is potentially dangerous. BE CAREFUL!! -- the RUIP MUST NOT allow system security to be compromised by that program. Implementing this feature may be more trouble than it is worth, since there are always bugs in operating systems, which could be exploited via this type of mechanism.

3.2.6. {U} ambiguity

Be aware that a malicious user's clever and/or persistent use of this feature can result in a list of most of the usernames on a system. Refusal of {U} ambiguity should be considered in the same vein as refusal of {C} requests (see section 3.2.2).

3.2.7. Audit trails

Implementations SHOULD allow system administrators to log Finger queries.

3.3. Client security

It is expected that there will normally be some client program that the user runs to query the initial RUIP. By default, this program SHOULD filter any unprintable data, leaving only the USASCII characters and CRLF. This is to protect against people playing with VT100 escape codes and changing other peoples' X window names, or committing other dastardly deeds. Two separate user options SHOULD be considered to modify this behavior, so that users may choose to view international data or control characters:

- one to allow characters less than ASCII 32
- another to allow characters greater than ASCII 126

For environments that live and breathe international data, the system administrator SHOULD be given a mechanism to enable these options by default for all users on a particular system. This can be done via a global environment variable or similar mechanism.

4. Examples

4.1. Example with a null command line ({C})

```
Site: elbereth.rutgers.edu
Command line: <CRLF>

Login       Name               TTY Idle    When            Office
rinehart Mark J. Rinehart      p0  1:11 Mon 12:15  019 Hill       x3166
greenfie Stephen J. Greenfiel  p1       Mon 15:46  542 Hill       x3074
rapatel  Rocky - Rakesh Patel  p3    4d Thu 00:58  028 Hill       x2287
pleasant Mel Pleasant          p4    3d Thu 21:32  019 Hill    908-932-
dphillip Dave Phillips         p5  021: Sun 18:24  265 Hill       x3792
dmk      David Katinsky        p6    2d Thu 14:11  028 Hill       x2492
cherniss Cary Cherniss         p7     5 Mon 15:42  127 Psychol    x2008
harnaga  Doug Harnaga          p8  2:01 Mon 10:15  055 Hill       x2351
brisco   Thomas P. Brisco      pe  2:09 Mon 13:37  h055           x2351
laidlaw  Angus Laidlaw         q0  1:55 Mon 11:26  E313C       648-5592
cje      Chris Jarocha-Ernst   q1     8 Mon 13:43  259 Hill       x2413
```

4.2. Example with name specified ({U}{C})

```
Site: dimacs.rutgers.edu
Command line: pirmann<CRLF>

Login name: pirmann                    In real life: David Pirmann
Office: 016 Hill, x2443                Home phone: 989-8482
Directory: /dimacs/u1/pirmann          Shell: /bin/tcsh
Last login Sat Jun 23 10:47 on ttyp0 from romulus.rutgers.
No unread mail
Project:
Plan:
                    Work Schedule, Summer 1990
                Rutgers LCSR Operations, 908-932-2443

                    Monday       5pm - 12am
                    Tuesday      5pm - 12am
                    Wednesday    9am -  5pm
                    Thursday     9am -  5pm
                    Saturday     9am -  5pm

                        larf larf hoo hoo
```

4.3. Example with ambiguous name specified ({U}{C})

```
Site: elbereth.rutgers.edu
Command line: ron<CRLF>
Login name: spinner                     In real life: Ron Spinner
Office: Ops Cubby,    x2443             Home phone: 463-7358
Directory: /u1/spinner                  Shell: /bin/tcsh
Last login Mon May  7 16:38 on ttyq7
Plan:
            ught i
          ca      n
        m           a
       ’       ...     t
      I      .   .     i
             !         m
      !       !       e
       p       !pool
        l
         e
          H


Login name: surak                       In real life: Ron Surak
Office: 000 OMB Dou,    x9256
Directory: /u2/surak                    Shell: /bin/tcsh
Last login Fri Jul 27 09:55 on ttyq3
No Plan.

Login name: etter                       In real life: Ron Etter
Directory: /u2/etter                    Shell: /bin/tcsh
Never logged in.
No Plan.
```

4.4. Example of query type {Q2} ({U}{H}{H}{C})

```
Site: dimacs.rutgers.edu
Command line: hedrick@math.rutgers.edu@pilot.njin.net<CRLF>

[pilot.njin.net]
[math.rutgers.edu]
Login name: hedrick                     In real life: Charles Hedrick
Office: 484 Hill, x3088
Directory: /math/u2/hedrick             Shell: /bin/tcsh
Last login Sun Jun 24 00:08 on ttyp1 from monster-gw.rutge
No unread mail
No Plan.
```

5. Acknowledgments

Thanks to everyone in the Internet Engineering Task Force for their comments. Special thanks to Steve Crocker for his security recommendations and prose.

6. Security Considerations

Security issues are discussed in Section 3.

7. Author's Address

David Paul Zimmerman
Center for Discrete Mathematics and
Theoretical Computer Science
Rutgers University
P.O. Box 1179
Piscataway, NJ 08855-1179

Phone: (908)932-4592

EMail: dpz@dimacs.rutgers.edu