GFD-R.xxx
FVGA-WG

Ralph Niederberger (Editor), Research Center Jülich

January 16, 2009

Firewall Traversal Protocol (FiTP)

Status of This Memo

This memo is a draft specification of the Firewall Traversal Protocol (FiTP). Distribution of this memo is unlimited.

Copyright Notice

Trademark

Abstract

Firewalls control traffic flows between internal and external communication partners. Mostly traffic from inside to outside is allowed, but traffic coming from outside must be explicitly configured. The rules which packets may traverse the firewall and which not are normally configured manually by firewall administrators. To speed up such kind of access list changes, it would be desirable to dynamically signal access requests and automatically change those access lists. Though some protocols are inspectable by firewalls already like FTP, SIP and H.323, a general protocol, which could be used for signaling dynamically required access rules, is not available until now.

This paper proposes a standard protocol, which would allow such signaling. Firewalls which have installed a corresponding inspection module could be automatically configured, which would ease the configuration of such systems a lot.

The proposed protocol (FiTP) can be used in two ways. First of all, a firewall aware of FiTP, could automatically allow connections signaled by authorized users. Secondly, an intermediate solution could be implemented, so that firewalls unaware of FiTP could be configured by the server process, which is the end point of the FiTP control connection. Via this approach a smooth transition would be possible. Installations having old firewall hard- and/or software could use the new protocol already, before installing a system which is FiTP enabled.

Contents

Introduction

Today's advantages of communication possibilities between sites worldwide using the global INTERNET imply vice versa severe risks to locally attached systems. Systems which are not secured adequately, could be hacked and misused or even data destroyed.

To protect local systems against security risks from outside, Firewalls are used. Firewalls control traffic flows between internal and external communication partners. Mostly traffic from inside to outside is allowed, but traffic coming from outside must be explicitly configured.

Normally configuring firewalls is manually done by firewall administrators. To speed up such kind of access list changes, some protocols, which use parallel sessions belonging to the same application, are known to firewall system software, so that dynamic connection attempts can be allowed though no explicit access rule had been available. One example is the well known FTP protocol, where allowed control connections have been specified within the access list of the firewall. Access rules for the data connection are built dynamically on the fly by inspecting the control connection. Data connections to be used will be signaled via the control connection, so that the firewall is able to recognize such requests.

Since many Grid applications are using also an unknown number of "n" ad-hoc parallel sessions, where "n" is dependent on the number of Grid nodes available, data streams needed, data servers available, etc, it would be desirable to have defined a standardized protocol, which allows to signal the need of those dynamic traffic flows to the firewall systems located on the way between source and destination.

The objective of FiTP is (1) to set up a secure authenticated connection between client and server over which (2) the need of dynamically required data connections can be signaled and (3) the firewalls located on the path between source and destination can change their local access lists accordingly.

It should be mentioned that the FiTP protocol can be used for signaling data connections only which use the same path as the control connection is using, since only firewalls which are on this path can read the control messages sent and are able to configure their access lists accordingly.

If the FiTP protocol extension is not implemented within a firewall, i.e. the firewall does not have implemented code being able to analyze FiTP messages, an intermediate solution could be to install a signaling subroutine on the server side, which allows setting access lists according to the access requests by CLI commands or proprietary software at the firewall.

This paper assumes knowledge of the Transmission Control Protocol, the SSH Transport Layer Protocol [SSH-TLP] and some background in X.509 certificates and encryption methods. Furthermore the reader should have some knowledge about Grid Applications and their issues with Firewalls [GFD-083] and [GFD-142].

1 Overview

In this section, the history, the terminology, and the FiTP model are discussed. The terms defined in this section are only those that have special significance in FiTP. Some of the terminology is very specific to the FiTP model; some readers may wish to turn to the section on the FiTP model while reviewing the terminology.

1.1 History

Grid-Projects with external partners lead to communication relationships between external and internal computer systems often requiring special configurations at firewall systems. These configurations include access for communication sessions (ports) and access to single systems or whole sub networks.

Only few firewall systems have been able to handle applications with dynamically assigned ports in the past. Some implementations exist for applications such as FTP, H.323, and SIP. But currently no general solution is available and explicitly no support for protocols used by Grid applications has been provided until now.

Often within a grid environment each institution or even worse each installation has its own firewall system. All of them have to be traversed by Grid applications. Because of the problems discussed above, project networks are placed in a demilitarized zone in most of the cases. This implies that every computer system used in the project has to be secured carefully. Wrongly configured systems lead to immediate security vulnerabilities.

Supercomputers, PC-clusters and/or special systems had to be connected via dedicated networks assuming a “Net of Trust”, since configuring access rules for those systems and their grid applications would be much to complicated or insecure, because of granted access which is not used for long periods. So compromises of these systems led to increased security problems.

The results of those special security requirements have been administrative overhead for deployment and protection of grid environments, wildcard access rights (ports not known, so access granted to whole system), weaker policies or no security policies at all, general decreasing the security level to that of the partner installation and security vulnerability because of open ports for long time periods.

So providing a standardized and widely-used control protocol allowing grid application to request access to internal resources in an easy to use, but secure manner is strongly desirable.

1.2 Definitions

access-rule

An access-rule prot,sDTH,sport1,sport2,dDTH,dport1,dport2 defines the privilege of a system or subnet sDTH using source ports within the port range [sport1,…,sport2] to transfer data to a system or subnet dDTH at port range [dport1,…,dport2] using the protocol prot (IP, TCP, UDP or IPSEC). The hosts sDTH and dDTH may be different to user-CH and auth-CH. If sDTH and/or dDTH are specifying subnets, then user-CH and/or auth_CH may be outside these subnets respectively. Using prot IP or IPSEC sport1, sport2, dport1 and dport2 are ignored.

auth-CH

The authentication and authorization control host (auth-CH) "listens" on Port L for a connection from a user host (user-CH). After connection by the user-CH the associated process auth-PI on host auth-CH authenticates the grid user and grants access to local grid resources. It receives standard FiTP commands from the user-PI and sends replies.

auth-PI

The authentication and authorization protocol interpreter listens at host auth_CH on port L for connections of a user on host user-CH, requesting access grants by FiTP commands for its application data streams.

control connection

A control connection is the communication path between the user-CH and the auth-CH for the exchange of commands and replies. This connection follows the SSH protocol but changes after key exchange and authentication to NONE cipher, i.e. a method to transfer data using the SSH protocol without any encipherment of the data, but leaving the Hashed Message Authentication Code (HMAC) routines in effect even for plaintext transfers.

data connection

A full duplex connection (TCP) over which data is transferred. For UDP transfers a virtual full duplex connection is assumed, as being two serial connections between sDTH and dDTH and using ports sport and dport. The data transferred itself between two DTHs is out of the scope of this document and may be a file transfer, message passing, etc.

data path grant

A data path grant is the access rule which has been included into the firewall configuration for a data connection granted via an FiTP control connection.

data port

The data transfer process "listens" on the data port for a connection from the active transfer process in order to open the data connection.

dDTH

The data transfer hosts are the source of the data connection (sDTH) and the destination of the connection (dDTH). sDTH establishes the connection to dDTH. If sDTH and/or dDTH are subnets, then the data transfer hosts are located within these subnets.

error recovery

Error recovery allows a user to recover from certain errors such as failure of either host system or transfer process. Within FiTP, error recovery may involve restarting a control connection process at a given checkpoint.

Firewall

A firewall is a logical object (hardware and/or software) within a network infrastructure which prevents communications forbidden by the security policy of an organization from taking place, analogous to the function of firewalls in building construction. Often a firewall is also referred to as a packet filter. The basic task of a firewall is to control traffic between different zones of trust and/or administrative authorities. Typical zones of trust include the Internet (a zone with no trust) and an internal network (a zone with high trust). The ultimate goal is to provide controlled connectivity between zones of differing trust levels through the enforcement of a security policy and a connectivity model based on the least privilege principle.

Proper configuration of firewalls demands skill from the administrator. It requires considerable understanding of network protocols and of computer security. Small mistakes can lead to a firewall configuration worthless as a security tool and, in extreme situations, fake security where no security at all is left.

FiTP commands

FiTP commands are a set of commands that comprise the control information flowing from the user-CH to the auth-CH host.

PI

The protocol interpreter is the software which implements the FiTP protocol in user-PI and auth-PI. The user and auth server sides of the protocol have distinct roles implemented in a user-PI and a server-PI.

reply

A reply is an acknowledgment (positive or negative) sent from the server to the user process via the control connection in response to FiTP commands. The general form of a reply is a completion code (including error codes) followed by a comma separated text string. The codes are for use by programs and the text is usually intended for human users.

sDTH

The data transfer host establishes the data connection with the "listening" data port. It sets up parameters for transfer and storage, and transfers data on command from its PI. The DTP can be placed in a “passive" state to listen for, rather than initiate a connection on the data port.

user

A person or a process on behalf of a person wishing to get dynamically opened a connection on a firewall for communication with an application server process.

user-CH

The user control host (user-CH) sends requests for port openings to Port L at the auth-CH. After positive processing of its requests, the user-CH starts the application processes which needed the port openings at the firewall.

user-PI

The user protocol interpreter initiates the control connection from its port U to the auth-CH server, initiates FiTP commands, and requests access grants for its application data streams.

1.3 The FiTP model

With the above definitions in mind, we can describe the FiTP model as follows:

The user-protocol interpreter (user_PI) initiates the control connection using the ssh protocol to a predefined special port for FiTP. The control connection follows the SSH protocol, but a modified version of SSH [SSHNCIPH] using NONE cipher encryption after the authentication and authorization phase will be used. After this initiation phase the user generates standard FiTP commands and transmits these to the server process via the control connection. Standard replies are sent from the server-PI to the user-PI over the control connection in response to the commands. Since the FiTP control connection uses “NONE cipher” encryption for the underlying ssh session, any message sent can be read by intermediate firewalls.

Since the Hashed Message Authentication Code (HMAC) routines are being used even for plaintext transfers, it is guaranteed that no man-in-the-middle can modify messages sent.

As described above authentication and authorization within FiTP is done via the surrounding ssh session. “grant access” commands allow requests for opening ports to be sent to the server side and being checked there for granting access.

FiTP allows the exchange of grant access requests and responses. For every request by the user_PI the server checks if he can grant this access to this user. So it is possible to provide specific users more privileges than others.

The server acknowledges the request and if needed configures the firewall accordingly. (This is only required, if the firewall has not already read and interpreted the user request, i.e. no FiTP inspection module for this firewall is available.) When the server has acknowledged an access rule, the user can start its data connection.

Several grant access requests can be issued by the user_PI in parallel to allow the signaling of multiple data connections with one FiTP control connection only.

The end of a data connection has to be signaled also, so that the access rules can be deleted from the firewall configuration again. Nevertheless, because of security reasons all access lists granted for an FiTP control connection will be deleted when the FiTP control connection, i.e. the encapsulation ssh session, has ended.

1.4 The Relationship between FiTP and SSH-NONE Cipher

FiTP uses a modification of the SSH protocol on the control connection. This can be achieved in two ways: first, the user-PI or the server-PI may implement the rules of the SSH-NONE Cipher Protocol directly in their own procedures; or, secondly, the user-PI or the server-PI may make use of the existing SSH module in the system.

Efficiency and independence argue for the first approach. Ease of implementation, sharing code, and modular programming argue for the second approach. In practice, FiTP relies on very little of the SSH Protocol, so the first approach does not necessarily involve a large amount of code.

Nevertheless it seems preferable to implement the user_PI as a software library, where key_exchange, authentication and authorization as well as grant access requests are provided as subroutines, so that any grid application can use these subroutines within their program codes. A sample command sequence is provided in chapter “A sample program using an FiTP software library” below.

1.5 Establishing data connections

The mechanics of transferring data over the granted communication paths is out of scope to this document. Any kind of application can be used, which use IP, TCP, UDP, or IPSEC transfer protocols.

Having granted access the communication path can be used. After the data transfer the user has to signal the end of the data communication to the auth_CH via the control connection so that the firewalls located on the path are aware of deleting access grants. Nevertheless the firewalls will close the data path grants just after the control connection has been ended.

1.6 Error recovery and restart

There is no provision for detecting bits lost or scrambled in data transfers; this level of error control has to be handled by TCP or the application itself.

However, a restart procedure is provided to protect users from gross system failures (including failures of a host, a FiTP-process, or the underlying network).

The restart procedure is defined only for the control connection here. Data connection errors have to be handled by the applications themselves.

Error recovery of the control connection may lead to loss of data path grants, which has to be handled by the applications also.

So error recovery is defined here only by means of restarting at predefined control status points.

Error recovery on client and server side is done by sending special restart control packets indicating the kind of error and point of restarting.

In case of disaster recovery any side can cancel the TCP session, so that all data path grants get destroyed. In case of disaster recovery the user has to restart from the beginning.

2 General remarks to FiTP commands and replies

The main purpose of the FiTP protocol is to inform firewalls located on the communication path of a desired data connection about the request for dynamic access rule configuration for this data traffic. To allow reading those requests on the fly it is necessary that all requests are sent in clear text. On the other way sending commands in clear text allows easy changing of this commands by a man-in-the-middle.

To ensure that commands are not modified on the communication path any command sent in clear text via the encapsulating ssh connection is protected by a Hash Message Authentication Code.

The communication channel from the user-PI to the auth-PI is established as a TCP connection from the user to the standard authentication and authorization server port. The user protocol interpreter is responsible for sending FiTP commands and interpreting the replies received; the auth-PI interprets commands, checks authentication and authorization and sends corresponding replies.

3 FiTP command and reply syntax and description

The Firewall Traversal protocol commands and replies consist of a four digit number followed by a phase describing text string and command parameters separated by commas. Every message is terminated by a <CRLF> character.

Replies have the same syntax, but have additionally an acknowledgement or negative acknowledgement (ACK, NACK) as third text string.

3.1 FiTP commands and replies

3.1.1 Grant access commands and replies

All grant access request and response commands are structured as follows:

[**4_digit_message_code**],**GAcR**,[textstring]

Responses have the following structure:

[**4_digit_message_code**],**GAcR**,[**ACK**|**NACK**],[textstring]

Valid commands are:

3000,**GAcR,Allow=***n*,*prot,h1,p1,p2,h2,p3,p4*

A user_PI requests access for an application which wants to communicate between ip addresses h1 and h2. The application uses the protocol *prot* and ports on the client side between p1 and p2 and on server side between p3 and p4. h1 and h2 may be single ip addresses or subnetworks. h1 and h2 have to be specified in modified CIDR notation with all quadruples using three digits and the CIDR block prefix using 2 digits (e.g. 192.168.016.000/22, which means all addresses within the class C networks 192.168.16.0, 192.168.17.0, 192.168.18.0 and 192.168.19.0). This allows easy scanning of the positional parameters by firewall hardware. “n” is a decimal number representing the n-th request, also being sent as 5 digit number. Furthermore ports have to be specified as 5 digit numbers (having 65536 a special meaning as “*”, i.e. all ports).

By specifying this decimal number this allows the auth_PI to send responses related to each individual grant request.

3000, **GacR,ACK,Allow=***n*,*prot*,*h1,p1,p2,h2,p3,p4*

Positive acknowledgement to the grant request n. For h1 and h2 specification see *3000 user_PI command above.*

3001,**GAcR,NACK,Form_error**

Auth_PI is awaiting a well formed 3000 or 3020 message, but got a malformed or quite different message.

3009,**GAcR,NACK,Deny,***n*

The access grant has been denied by the auth_PI

3020,**GAcR,Close=***n,prot***,***h1,p1,p2,h2,p3,p4*

The access grant will not be needed anymore. It can be discarded. For h1 and h2 specification see *3000 user_PI command above.*

3020,**GAcR,ACK,Close=***n,prot***,***h1,p1,p2,h2,p3,p4*

This is a positive feedback from auth_PI. Access grants will be destroyed. For h1 and h2 specification see *3000 user_PI command above.*

3021,**GAcR,NACK,*Form error***

Auth_PI is waiting for a 3000 or 3020 message, but got a malformed or quite different message.

3022,**GAcR,NACK,***n***,NoConnection**

User_PI has requested to delete an access grant, which is not known to the auth_PI. This may be related to a former timeout for this access grant or misconfiguration.

3600,**GacR,Allow6**=*n,prot,h6-1,p1,p2,h6-2,p3,p4*

IPv6 version of 3000 message. See “3000, GacR,Allow=...”. h6-1 and h6-2 are the IPv6 addresses of source and destination hosts/nets of the data connections for which the firewall should allow access.

3600,**GacR,ACK,Allow6=***n,prot***,***h6-1,p1,p2,h6-2,p3,p4*

IPv6 version of server response 3000 message

3601,**GAcR,NACK,Form_error**

IPv6 version of server negative response 3001 message

3609,**GAcR,NACK,Deny6,**n

IPv6 version of server deny 3009 message

3620,**GAcR,Close6=***n,prot***,***h6-1,p1,p2,h6-2,p3,p4*

IPv6 version of client close request 3020 message

3620,**GAcR,ACK,Close6=***n,prot***,***h6-1,p1,p2,h6-2,p3,p4*

IPv6 version of positive server close response 3020 message

3621,**GAcR,NACK,*Form error***

IPv6 version of server response 3021 message

3622,**GAcR,NACK,**{n},**NoConnection**

IPv6 version of server response 3022 message

3.1.2 Control Session commands and replies

All control sessions commands are structured as follows:

[**4_digit_message_code**],**PathCntl**,[textstring]

Responses have the following structure:

[**4_digit_message_code**], **PathCntl**,[**ACK**|**NACK**],[textstring]

Valid commands are:

0000,**PathCntl**

This command is send as the first command after the ssh session has been established, i.e. TCP-Hand-Shake done, authentication and authorization checked and cipher changed to NONE cipher.

0000,**PathCntl,ACK**

This is the positive answer of the authentication server protocol interpreter auth_PI to the 0000 control session initiation command of the user_PI. Auth_PI is ready for session.

0001,**PathCntl,NOOP**

This command may be sent by user_PI process at any time to check if the server is still available. This can be used e.g. to hold up the control connection when a timeout would appear at the remote side otherwise.

0001,**PathCntl,NOOP,ACK**

This is the acknowledgement to a previous sent 0001 message by the user_PI client process.

0002,**PathCntl,NOOP**

This command may be sent by auth_PI process at any time to check if the client is still available. This can be used e.g. to hold up the control connection when a timeout would appear at the remote side otherwise.

0002,**PathCntl,NOOP,ACK**

This is the acknowledgement to a previous sent 0002 message by the auth_PI server process.

8000,**PathCntl**

This command is sent by user_PI process for finalizing control connection.

8000,**PathCntl,ACK**

Positive response of the auth_PI to the finalizing request 9000 by user_PI.

9000,**PathCntl**

user_PI or auth_PI have diagnosed a severe problem and will kill the control connection immediately. The PI will not wait for any response. This message may be sent from both sides at any time when identifying any abnormalities.

4 FiTP replies

4.1 Structure of FiTP reply codes

Replies to Firewall Traversal Protocol commands are devised to ensure the synchronization of requests to guarantee that the user process always knows the state of the server. Every command must generate at least one reply. The replies show the existence of an intermediate state if all preceding commands have been successful. A failure at any point in the sequence necessitates the repetition of the entire sequence from the beginning.

The details of the command-reply sequence are made explicit in a set of state diagrams below.

An FiTP reply consists of a four digit number (transmitted as four alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is intended for the human user. It is intended that the four digits contain enough encoded information that the user-process (the User-PI) will not need to examine the text and may either discard it or pass it on to the user, as appropriate.

In some cases the user_PI has to correlate responses to earlier sent requests (requesting several grant access rules with a request number included (see the “n” within the 3009, 3020 and 3021 responses).

The four digits of the reply each have a special significance. This is intended to allow a range of very simple to very sophisticated responses by the user-process. The first digit denotes which phase of the control connection has been reached. Digits two and three denote states within those control connection phases. Digit four denotes completion of requests or negative acknowledgements. There are five values for the first digit of the reply code:

0xyz

The control session has just been started no further commands have been interchanged.

1xyz

Key exchange phase has started and is in progress.

2xyz

Authentication and authorization phase is in progress.

3xyz

Grant access commands and replies are being sent.

8xyz

The control connection is in the closing process.

9xyz

user_PI or auth_PI have diagnosed a severe problem and will kill the control connection immediately.

Digit two is phase dependent. For phase 3, i.e. grant access commands, it differentiates between IPv4 and IPv6 commands having x=4 or x=6 respectively.

Digit three is also very phase dependent and corresponds to the different states the communication between client and server has entered.

Digit four within response messages are defined as follows:

xyz0

This indicates a positive response. The auth_PI acknowledges that it has received the request correctly and has changed its state accordingly. For grant access requests it acknowledges that the request has been accepted.

xyz1

The auth_PI has received a message from the user_PI which is illegally formatted, out of sequence or corrupted.

xyz{n}

This is a negative response. The previous request cannot be acknowledged. {n} may very from 2 to 8.

xyz9

This is a negative response. The previous request cannot be granted. Dependent on the phase entered this may need starting this phase from the beginning or a grant access request has to respecified (e.g. access to a special IP address is not allowed, but to another one an access could be granted.)

4.2 Numeric Order List of Reply Codes

0000,**PathCntl**

0000,**PathCntl,ACK**

0001,**PathCntl,NOOP**

0001,**PathCntl,NOOP,ACK**

0002,**PathCntl,NOOP**

0002,**PathCntl,NOOP,ACK**

3000,**GAcR,Allow**=*n,prot,h1,p1,p2,h2,p3,p4*,

3000,**GAcR,ACK,Allow=***n,prot***,***h1,p1,p2,h2,p3,p4*

3001,**GAcR,NACK,Form_error**

3009,**GAcR,NACK,Deny,**n

3020,**GAcR,Close=***n,prot***,***h1,p1,p2,h2,p3,p4*

3020,**GAcR,ACK,Close=***n,prot***,***h1,p1,p2,h2,p3,p4*

3021,**GAcR,NACK,*Form error***

3022,**GAcR,NACK,**{n},**NoConnection**

3600,**GAcR,Allow6**=*n,prot,h6-1,p1,p2,h6-2,p3,p4*

3600,**GAcR,ACK,Allow6=***n,prot***,***h6-1,p1,p2,h6-2,p3,p4*

3601,**GAcR,NACK,Form_error**

3609,**GAcR,NACK,Deny6,**n

3620,**GAcR,Close6=***n,prot***,***h6-1,p1,p2,h6-2,p3,p4*

3620,**GAcR,ACK,Close6=***n,prot***,***h6-1,p1,p2,h6-2,p3,p4*

3621,**GAcR,NACK,*Form error***

3622,**GAcR,NACK,**{n},**NoConnection**

8000,**PathCntl**

8000,**PathCntl,ACK**

9000,**PathCntl**

Declarative specifications

4.3 Connections

The server protocol interpreter shall "listen" on Port L. The user or user protocol interpreter shall initiate the full-duplex control connection. Server and user processes should follow the conventions of the SSH protocol. After connection establishment and authentication and authorization checks the connection should change to NONE cipher encryption using HMAC enabled for protection against man-in-the-middle attacks. The control connection shall be closed by the server at the user's request after all transfers and replies are completed.

The data connection will be initiated from host(s) sDTH to the server(s) running at dDTH (host or subnet). The data connection is out of scope of this document allowing any kind of TCP or “virtual” UDP, IP or IPSec communication.

If at any time either the user or server_PI observes that the connection is being closed by the other side, it should promptly read any remaining data queued on the connection and issue the close on its own side.

4.4 Commands

The commands are text strings transmitted over the control connections as described in the section on FiTP Commands.

The command syntax is specified here.

The commands begin with a four digit alphanumeric code followed by a human readable phase descriptor and an argument field. Upper and lower case alphabetic characters are to be treated identically. This also applies to any symbols representing parameter values such as the key exchange method text string (e.g. certificate). Thus, any of the following may represent a grant access request command:

3000,GAcR,ACK,Allow=*00020,TCP*,123.045.067.089/32*,12345,12359,124.111/32.222.233,05000,05010*

3000,gacr,ack,allow=*00020,tcp*,123.045.067.089/32*,12345,12359,124.111/32.222.233,05000,05010*

3000,GACR,ACK,ALLOW=*00020,TCP*,123.045.067.089/32*,12345,12359,124.111/32.222.233,05000,05010*

3000,Gacr,Ack,Allow=*00020,Tcp*,123.045.067.089/32*,12345,12359,124.111/32.222.233,05000,05010*

The argument field consists of a variable length character string ending with the ASCII <CRLF>. Any text string following x”00” will be ignored and deleted.

The syntax is specified below in 8bit-ASCII. All characters in the argument field are ASCII characters including any ASCII represented decimal integers. Square brackets

denote an optional argument field. If the option is not taken, the appropriate default is implied.

4.5 FiTP command options

The following optional fields are allowed within FiTP commands:

Allow= <five-digit-integer> <,> <*prot*> <,>
<*ip-cidr*> <,> <*port*> <,> <*port*> <,>
<*ip-cidr*> <,> <*port*> <,> <*port*> <CRLF >
Close= <five-digit-integer> <,> <*prot*> <,>
<*ip-cidr*> <,> <*port*> <,> <*port*> <,>
<ip-cidr> <,> <port> <,> <port> <CRLF >
Allow6= <five-digit-integer> <,> <*prot*> <,>
<*ipv6-cidr*> <,> <*port*> <,> <*port*> <,>
<*ipv6-cidr*> <,> <*port*> <,> <*port*> <CRLF >
Close6= <five-digit-integer> <,> <*prot*> <,>
<*ipv6-cidr*> <,> <*port*> <,> <*port*> <,>
<ipv6-cidr> <,> <port> <,> <port> <CRLF >

4.6 FiTP command option syntax

The syntax of the above option fields (using BNF notation where applicable) is:

<five-digit-integer> ::= < “00001” | “00002” | … | “65535” >
i.e. any integer 1 through 65535 specified in 5 digits
<prot> ::= <”IP”> | <”TCP”> | <”UDP”> | <”IPSEC”>
<ip-cidr> ::= <ipaddr><”/”><netprefix>
<ipv6-cidr> ::= <ipv6addr><”/”><v6netprefix>
<ipaddr> ::= <3-digit-qual><.><3-digit-qual><.><3-digit-qual><.><3-digit-qual>
<3-digit-qual> ::= < “000” | “001” | ... | “254” | “255” >
i.e. any integer 0 through 255 specified in 3 digits
<netprefix> ::= < “00” | “01” | … | “32” >

<ipv6addr> ::= <hex4><”:”><hex4><”:”><hex4><”:”><hex4><”:”>
<hex4><”:”><hex4><”:”><hex4><”:”><hex4>
i.e. any decimal integer 0 through 32 specified in two digits
<hex4> ::= <hex><hex><hex><hex>
i.e. any 4 character hex string
<hex> ::= <”0”> | <”1”> | <”2”> | <”3”> | <”4”> | <”5”> | <”6”> | <”7”> | <”8”> |
<”9”> | <”A”> | <”B”> | <”C”> | <”D”> | <”E”> | <”F”>
i.e. any hex character 0 through F
<v6netprefix> ::= < “000” | “001” | … | “128” >
<port> ::= <five-digit-integer> | <”65536”> i.e. any decimal integer 1 through 65536

4.7 Sequencing of commands and replies

The communication between user and server is intended to be an alternating dialogue. As such, the user issues a FiTP command and the server responds with a prompt primary reply. The user should wait for this initial primary success or failure response before sending further commands.

Spontaneous Replies

Sometimes "the system" spontaneously has a message to be sent to a user (usually all users), for example, "System going down in 15 minutes". User_PI and/or auth_PI may send a

9000,**PathCntl**

command. Receiving such a message, both sides should immediately close the control connection to be sure that the firewalls on the path are aware of such problems and are able to delete access rights granted for those sessions.

Command-Reply Sequences

In this section, the command-reply sequence is presented. Each command is listed with its possible replies; command groups are listed together. This listing forms the basis for the state diagrams, which will be presented separately.

Connection Establishment
0000

0000

Grant Access Request and Responses for IPv4

Grant Access (Allow=)

3000

3000,3001,3009

Closing of Access Grants (Close=)

3020

3020,3021,3022

Grant Access Request and Responses for IPv6

Grant Access (Allow6=)

3600

3600,3601,3609

Closing of Access Grants (Close6=)

3620

3620,3621,3622

Connection Closing

8000

8000

Abnormal Closing of connections

9000

9000

Keep alive testing

Client initiated

0001

0001

Server initiated

0002

0002

5 State diagrams

Here we present the state diagrams of client and server for an FiTP implementation. The diagrams use boxes labled with e.g. “Sxxxx” (entering state Sxxxx, where the next action

will be sending a message) and “WxxxxR” (waiting for a reply by the corresponding side; timeout will be 60 seconds). If no response arrives within 60 seconds an “xxx9” reply is assumed. Some boxes contain explicitly what will be done entering this box. Sometimes choices (Yes or No) decide which program path has to be followed. These choices are dependent on the code where the FiTP client library routines are called from. These choices are not part of the FiTP specification.

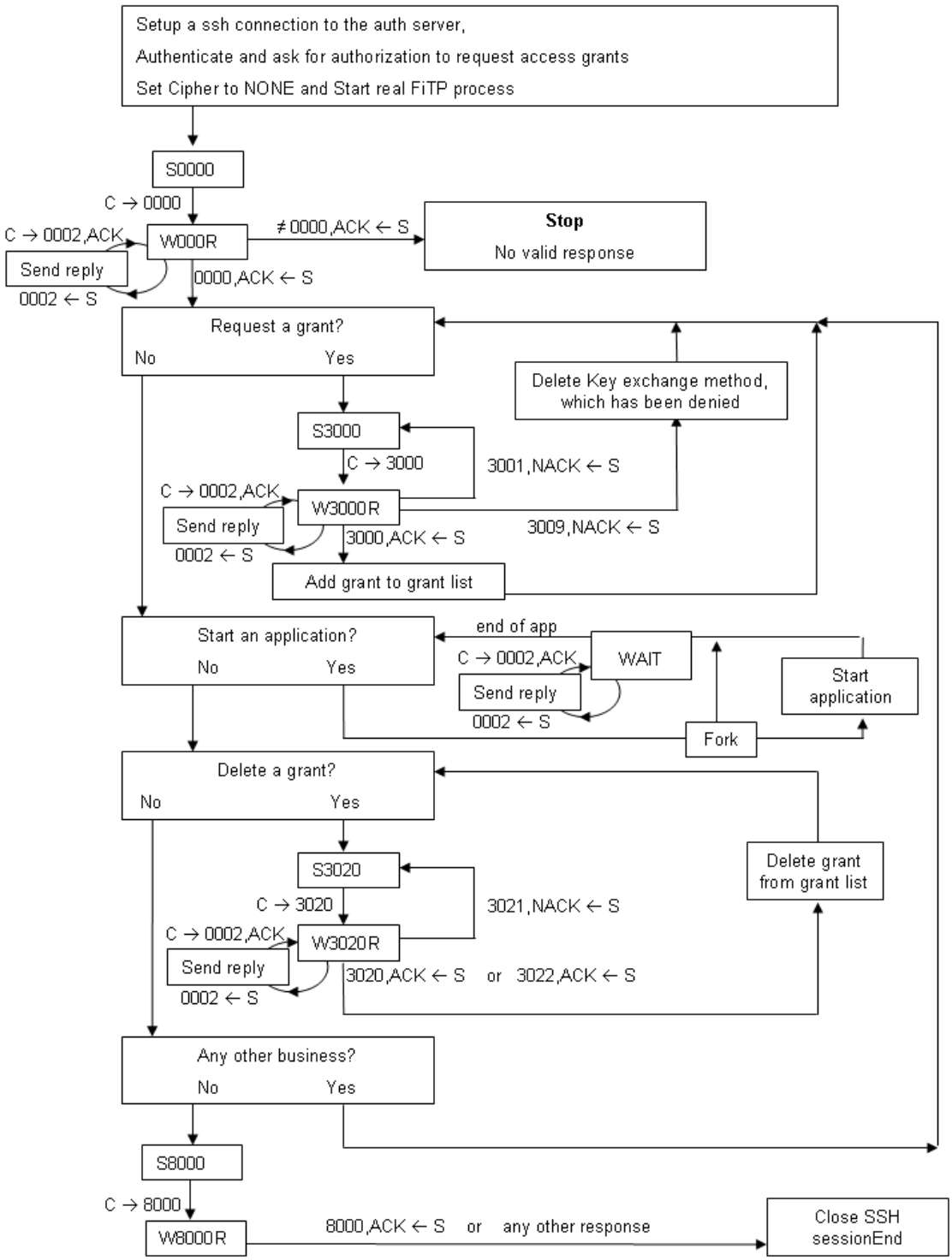
The following text strings within these diagrams have the following meaning:

C → xxxx	client sends an FiTP command with message code "xxxx"
S → xxxx	server replies to an FiTP command with message code "xxxx"
xxxx ← C	a message request has been received from the client.
xxxx,(N)ACK ← S	a message/reply has been received from the server. The command/reply code received is "xxxx". ACK (acknowledgement), NACK (negative acknowledgement)

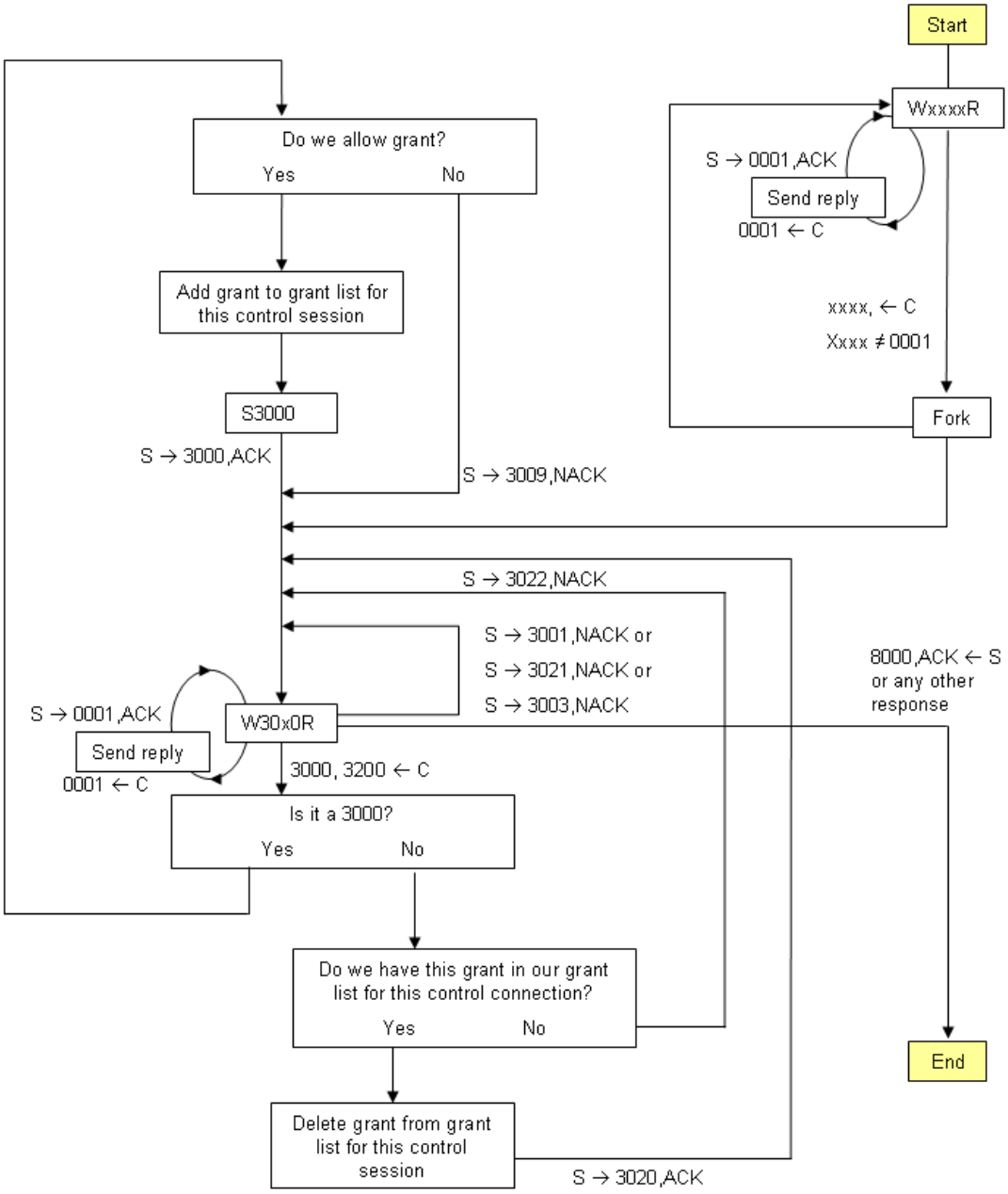
First the state diagram for the client is shown, after this the state diagram of the server is sketched. Within Grant Access commands the state diagrams for IPv4 are shown only. The state diagrams for IPv6 are accordingly. So mixing IPv4 and IPv6 commands and replies will lead to FormError messages, because a 30xx command cannot be answered by a 36xx response and wise versa.

It should be mentioned here, that the server has to be written in this way, that several clients can connect to the server in parallel asking for port opening requests. The state diagram for the server shows the states for a single client connection only.

State diagram for the FiTP client



State diagram for the FiTP server



6 A typical FiTP scenario

A User at host “U” wanting to use a communication path between host “S” port “sp” and host “D” port “dp” connects to the remote authentication and authorization server “AAS” at port “L”: In general, the user will communicate to the server “AAS” via this control connection his request to use this communication points. Server “AAS” authenticates the user and checks for his authorization to start communication between these endpoints. All firewalls on this control path are able to read the requested grants and are able to dynamically open the required ports in their access lists. The following message exchanges may be a typical scenario. '---->' represents commands from host U to host AAS, and '<----' represents replies from host AAS to host U.

[FW]

[A] ← **--- --- --- ---** → **[B]**

###

SSH session initiation

Authentication and authorization done

Cipher set to NONE

##

###

Control session initiation

###

→ SSH: 0000,PathCntl

← SSH: 0000,PathCntl,ACK

###

Authentication valid. Now starting Grant Access Request Path

##

Requesting access for GridFTP control connection

from host 192.168.15.116 i.e. netprefix 32

to host 172.100.14.123 (again netprefix 32)

source port 30123, destination port 2811

###

→ SSH: 3000,GAcR,Allow=
00001,TCP,192.168.015.116/32,30123,30123,172.100.014.123/20,02811,02811

← SSH: 3000,GAcR,ACK,Allow=
00001,TCP,192.168.015.116/32,30123,30123,172.100.014.123/20,02811,02811

###

Requesting access rules for 50 data connection for the above GridFTP session
from host 192.168.15.116 i.e. netprefix 32
to host 172.100.14.123 (again netprefix 32)
lower source port 30124 and upper source port 30173,
lower dest port 20001 and upper destination port 20050,
i.e. we want to connect from ports out of the port range [30124, … , 30173]
to ports within the range of [20001, … , 20050]
##
The request setup below would allow any combination of port combination
within the specified port ranges,
e.g. from port 30150 to port 20002 and from port 30125 to port 20014
###
→ SSH: 3000,GAcR,Allow=
00002,TCP,192.168.015.116/32,30124,30173,172.100.014.123/32,20001,20050
→ SSH: 3000,GAcR,ACK,Allow=
00002,TCP,192.168.015.116/32,30124,30173,172.100.014.123/32,20001,20050
###
Now starting a GridFTP process between 192.168.15.116 and 172.100.14.123
##
This part is not shown here.
….
##
After GridFTP has completed, we have to delete the access grants
###
Delete grants for Gridftp control connection
###
→ SSH: 3000,GAcR,Close=
00001,TCP,192.168.015.116/32,30123,30123,172.100.014.123/20,02811,02811
← SSH: 3000,GAcR,ACK,Close=
00001,TCP,192.168.015.116/32,30123,30123,172.100.014.123/20,02811,02811
###
Delete grants for GridFTP data connections
###
→ SSH: 3000,GAcR,Close=
00002,TCP,192.168.015.116/32,30124,30173,172.100.014.123/32,20001,20050

← SSH: 3000,GAcR,ACK,Close=
00002,TCP,192.168.015.116/32,30124,30173,172.100.014.123/32,20001,20050

###

All requested access rules have been deleted. Now closing control connection

###

→ SSH: 8000,PathCntl

← SSH: 8000,PathCntl

###

Closing SSH session

###

All done.

###

7 The typical FiTP scenario shown as sample program using a FiTP software library

```
#!/usr/bin/perl
#
# this is a sample perl program using FiTP software library calls for dynamic opening of firewalls
#
################################################################################
use strict;
use Socket;
use fitp;

my $VERSION = "FiTP_v_2.4";
my $authsrvip;
my authsrvport = “4711”;
my $ret;
my $timeout;
my $timeoutreq;
my $status;
my $remstatus;

my $packed_ip = gethostbyname("authserver.at.remotesite.com");
if (defined $packed_ip) {
    $authsrvip = inet_ntoa($packed_ip);
}
my $paddr; # save socket structure
my $proto; # save protocol number for tcp

$paddr  = sockaddr_in($authsrvport, $ipacked_ip);
$proto  = gethostbyname('tcp');
socket(SOCK, PF_INET, SOCK_STREAM, $proto) || die "socket: $!";
connect(SOCK, $paddr) || die "connect: $!";

start_key_exchange();
start_user_authenticate();

$ret=fitp::dyna_conn_open4(“00002”,”TCP”,”192.168.015.116/32”,”30124”,”30173”,
                                ”172.100.014.123/32”,”20001”,”20050”);
```

```
if ($ret == 0) {
    call_applications_using_opened_port();
    while (wait_for_applications_finished_or_timeout() != 0 ) {;
        if ( $timeout ) {
            $ret=fitp::timerefresh($status);
        }
        elsif ( $timeoutreq ) {
            $ret=fitp::answer_timeoutrequest($remstatus);
        }
        else {
            print “Unknown interrupt. Died”;
            exit 98;
        }
    }
}
else {
    print “Dynamic opening not allowed or error when requesting”;
    exit 99;
}
$ret=fitp::dyna_conn_close4(“0002”);
$ret=fitp::closecontrl_session();
exit;
}
```

8 Connection establishment

The FiTP control connection is tunneled via a SSH connection using NONE ciphers. It is established via TCP between the user process port U and the server process port L. This protocol is assigned the service port 4711, that is L=4711. (Instead of 4711 an official port number should be used and agreed on later)

9 Design alternatives

The FiTP model has been designed to support a variety of security scenarios.

The first scenario assumes that the firewalls to be dynamically configured already have code integrated which is aware of the FiTP protocol, so that the firewalls are able to configure automatically the requested access rules. This scenario also includes asymmetric routing as long as both paths, source to destination and destination to source, are traversing all firewalls, which have to be dynamically configured. The scenario also assumes that the data connections specified by the quadruples source-ip, source-port, dest-ip and dest-port are also traversing these firewalls in both directions.

The second scenario assumes that the firewall is not aware of FiTP and the auth-server is able to request the firewall to dynamically reconfigure its access tables. This could be done via CLI commands or special firewall device dependent software interfaces for configuration. This would imply that every auth-server has access to those configuration routines.

A third approach would have a special firewall agent available, which can be contacted from every auth-server. This firewall agent would check if the auth-servers are allowed to request reconfiguration of the firewall and would then be the only one allowed to actively reconfigure the firewall.

Though scenario one is intended to be the primary way for the FiTP protocol usage, the other alternative scenarios will work also. Additionally those scenarios allow configuration of firewalls, which are not located on the control path, but on the data paths only.

Within these different scenarios security policies can be realized manifold.

The firewall can be configured to allow FiTP control connections to a predefined number of auth-servers only. Furthermore it can be configured to limit dynamical opening of ports to specific ip addresses only, dependent on the auth-server, which has authorized these requests (e.g. it could be configured to allow port requests only, denying any request for opening of port ranges). Also a maximum number of parallel data connections allowed could be fixed. There are many other restrictions you could think of.

The same restrictions can be introduced using the alternative scenarios above. There could be a hierarchy of authentication, where the predefined firewall rules are highest rated. The firewall agent could further restrict access dependent on the auth-server who requests data paths. The auth-server itself could further restrict the rules which would be allowed by the firewall agent dependent on the client requesting data connections.

10 Summary

The FiTP protocol described below will provide grid applications with a tool for easy opening of firewall ports.

The past has shown that access rules for grid applications will be configured into firewalls for long period though they are used only within short time period. These constantly open ports are a potential security risk which can be minimized when those rules are configured only in time periods where they are really needed.

Usage of the protocol within grid applications minimizes the time period in which traffic can pass the firewall. The other way round it allows opening of ports without manual interaction of a firewall administrator. This leads to fast configuration independent of availability of those administrators. The authenticated and authorized interaction

between user applications and authentication/authorization servers provides a secure configuration of the firewalls involved.

It is recommended to use this protocol widely.

11 Security Considerations

This FiTP protocol allows easy dynamic configuration of firewall systems by authorized, but external users. This introduces a potential security risk, which will be analyzed here.

If we think about a normal IPsec communication, which we have allowed to traverse our firewall, security is assumed to be handled by the receiving server process. If we can assume, that the server system has not been hacked, we can postulate, that the authentication and authorization of the remote user has been checked and that he is allowed to use this data connection. If we assume furthermore that the receiving side allows packet forwarding to internal hosts, we have allowed connectivity to any internal host and port via this tunneling technique and only relying on the security checks done by the server system.

The same principle applies to FiTP. We allow control connections to a number of internal servers accessible from remote systems because of access rules within our firewall. Via dynamic requests the external user can ask for access to hosts inside of our organization normally protected by our organizational firewall. We again assume that the internal server has checked authentication and authorization, so that we can postulate that the remote user is allowed to ask for those port openings. The main difference to the model above is, that we now exactly know, where the communication streams are going. Since there is no tunneling of data connections anymore, we are aware of any communication possible. (Of course, the user could again use tunneling techniques on the connections he has got opened.

So the main security difference is the FiTP protocol itself. The main question arises: Can we trust this protocol?

The protocol uses common techniques for secure communication between partners.

- It uses the well known ssh protocol for encapsulation of FiTP commands.
- It uses common authentication and authorization techniques to check if the remote user is allowed to request port openings via SSH. This authentication and authorization process is encrypted, so that a man-in-the-middle cannot intercept and modify messages exchanged. Exchange of FiTP commands in clear text can only be started after this authentication/authorization.
- And it requests grants, though publicly readable, but protected via the HMAC, routines enabled within the encapsulating SSH connection. So any message sent from the corresponding side can be checked by the recipient for any modifications done by man-in-the-middle hackers. If any anomalies arise, both sides of the FiTP control stream may stop the communication and cancel the encapsulating SSH session. This policy is a strict implementation of the firewall rule: If something is going wrong, do not allow any further access. It is better to allow nothing than allowing everything.

12 Acknowledgements

The authors whish to thank Jan Mönnich from DFN-Cert for pointing us to the SSH NONE cipher solution, which helped save a lot of work concerning key exchange and authentication and authorization. ……**has to be updated**……. for feedback and comments on the document as well as for proof reading, corrections of spelling, grammar and style. We also would like to acknowledge the presentations from researchers in the OGF FI-RG and FVGA-WG sessions that helped shape this document.

Also we would like to thank the OGF security area director David Groep and infrastructure directors Richard Hughes-Jones and Cees de Laat for supporting our work.

13 Contributors

Ralph Niederberger (Editor)
Forschungszentrum Jülich GmbH
r.niederberger@fz-juelich.de

…others to be added…

14 Glossary

CIDR	Classless Inter Domain Routing (CIDR) is a method for assigning IP addresses without using the standard IP address classes like Class A, Class B or Class C. In CIDR, depending on the number of hosts present in a network, IP addresses are assigned. In CIDR notation, an IP address is represented as A.B.C.D /n, where "/n" is called the IP prefix or network prefix. The IP prefix identifies the number of significant bits used to identify a network. For example, 192.9.205.22 /18 means, the first 18 bits are used to represent the network and the remaining 14 bits are used to identify hosts.
GridFTP	Special FTP protocol for Grids which allows transferring a file via multiple parallel data sessions.
H.323	H.323 is an umbrella recommendation from the ITU-T that defines the protocols to provide audio-visual communication sessions on any packet-switched network.
HMAC	Keyed-hash message authentication code is a special Message Authentication Code (MAC) used on several protocols like TLS and IPsec and being based on crypthographic hash functions.
IPSec	IP Security, a set of protocols developed by the IETF to support secure exchange of packets at the IP layer. IPsec has been deployed

	widely to implement Virtual Private Networks (VPNs).
SIP	Session Initiation Protocol. It is an application-layer control protocol that can establish, modify, and terminate multimedia sessions such as Internet telephony calls (VoIP). SIP can also invite participants to already existing sessions, as in multicast conferences. Media can be added to (and removed from) an existing session. SIP transparently supports name mapping and redirection services, which supports personal mobility - users can maintain a single externally visible identifier regardless of their network location. See also RFC 3261, 3262, 3263, 3264, and 3265.
X.509	X.509 is a widely used standard for digital certificates.

15 Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

16 Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

17 Full Copyright Notice

be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the OGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

18 References

[GFD-083] Niederberger,R. (Editor) Fiirewall Issues Overview, Open Grid Forum, Oct. 2006, http://www.ogf.org/documents/GFD.83.pdf

[GFD-142] Metsch,T. (Editor), Requirements on operating Grids in Firewalled Environments, , Open Grid Forum, Oct. 2008, http://www.ogf.org/documents/GFD.142.pdf

[SSHNCIPH] HPN SSH/SCP NONE Cipher Switching, Chris Rapier PSC, Michael Stevens CMU, http://www.psc.edu/networking/projects/hpn-ssh/none.php

SSH-TLP Ylonen, C.Lonvick, RFC 4253, The Secure Shell (SSH) Transport Layer Protocol, January 2006, http://www.ietf.org/rfc/rfc4253.txt