# FiTP - A protocol draft for dynamic opening of Firewalls

r.niederberger@fz-juelich.de

# Overview

- Introduction
- From FTP(File Transfer Protocol)
  to FiTP (Firewall Traversal Protocol)
- SSH None Cipher switching coming along
- Future Extension requests
  e.g. “Token Based Firewaling approach”
- Summary

# Introduction

- Grids are becoming more and more familiar and widely used.
- But global INTERNET access is risky and should be secured adequately.

- Firewalls control traffic flows between internal and external communication partners.
- Especially traffic coming from outside should be explicitly filtered.
- Rules, which packets may traverse the firewall, are normally configured manually.
- Dynamic opening would speedup such processes.
- Some protocols like FTP, SIP and H.323, support dynamical configuration.
- A widely usable solution should be simple, easy to be implemented and to be used.
- Allowing an intermediate solution would be helpful.

# From FTP to FiTP (1)

- Most firewalls are aware of the FTP protocol.
- FTP is implemented using two streams
  - a control stream and a data stream
- Unfortunately FTP has some drawbacks:
  - Authentication is done via cleartext
  - it is not explicitly specified, if multiple data streams in parallel are allowed.
  - so most Firewalls support only serialized data streams

- But Grid applications demand multiple streams, e.g. for high throughput
- Often Grid applications use 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.

# From FTP to FiTP (2)

**Checking RFC 959 “File Transfer Protocol”, leads to a simple solution:**

- **Delete all things we don’t need.**
- **Change things we don’t like.**
- **Add things which are needed additionally.**
- **Have security in mind all the time.**

- **The outcome is a new protocol which is simple, easy to implement and widely usable.**
- **If defined as a standard, it will become commonly used and widely implemented.**

# From FTP to FiTP (3)

**The objective of FiTP is**

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

**Note:**

**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 configure their access lists accordingly.**

# From FTP to FiTP (4)

FiTP will not be available in Firewall Systems just after definition of the standard.

So there will be firewalls which don't have implemented code being able to analyze FiTP messages.

An intermediate solution has to be developed.

Installation of a signaling subroutine on the server side could setup access lists according to the access requests by CLI commands or proprietary software at the firewall.

Additional benefit could be:

This routine could signal requests to firewalls not located on the control path.

# From FTP to FiTP (5)

**Requirements:**

| | | |
|---|---|---|
| Message Integrity | Encrypted | Key exchange |
| Message Integrity | Encrypted | Authentication |
| Message Integrity | Encrypted | Authorization |
| Message Integrity | Clear Text | Control Information |
| Message Integrity | Clear Text | Acknowledgment |
| - | - | Data Connection (Out of scope) |
| Message Integrity | Clear Text | End of Service Signaling |

**SSH would provide most of this, but control info could not be check by intermediate firewalls.**

# SSH None Cipher Switching at a glance

**HPN SSH/SCP NONE Cipher Switching**

**by Chris Rapier PSC, Michael Stevens CMU, email: hpn-ssh@psc.edu**

- **NONE cipher is a method to transfer data using the SSH protocol without any encipherment of data which drastically decreased load on the CPU.**
- **Encipherment not necessary, when binary data or data already encrypted.**
- **Requirements of high speed transfers led to method using NONE cipher.**
- **If it is possible to have full encryption of the authentication process and switching the cipher to NONE afterwards, needs of the users and the system administrators will be met.**
- **Most of the mechanisms needed already in place in OpenSSH**
- **A method is used through which a message will be passed between the client and server requesting a cipher switch.**
- **Then advertisement that NONE cipher is the only available cipher (assumed both sides allow NONE cipher)**
- **To assure data integrity (man in the middle attacks), the Hashed Message Authentication Code (HMAC) routines have been left in place.**
- **Anytime SSH spans a TTY or if the user uses the '-T' to supress TTYs the NONE cipher is disabled.**

# The FiTP model (1)

- Client initiates the control connection using the ssh protocol to a predefined special port for FiTP.
- The control connection follows modified SSH protocol using NONE cipher encryption.
- After initiation phase the user generates standard FiTP commands and transmits these to the server process via the control connection.
- Standard replies are sent from server to client in response to the commands.
- Since the FiTP control connection uses “NONE cipher” encryption for underlying ssh session, any message sent can be read by intermediate firewalls.
- Since the Hashed Message Authentication Code (HMAC) routines are used, 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.

# The FiTP model (2)

- "grant access" commands allow requests for opening ports to be sent to the server side and being checked there for granting access.
- FiTP allows exchange of grant access requests and responses.
- For every request by the client 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. (Only required, if firewall is FiTP unaware)
- After server has acknowledged an access rule, the user can start its data connection.
- Several grant access requests can be issued by the a client 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 a FiTP control connection will be deleted when the FiTP control connection, i.e. the encapsulation ssh session has ended.

# FiTP & SSH-NONE Cipher relationship

FiTP uses a modification of the SSH protocol on the control connection.

This can be achieved in two ways:

- first, client and server may implement the rules of the SSH-NONE Cipher Protocol directly in their own procedures or
- secondly, client and server 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 client 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 within their program codes.

# FiTP commands & replies Control Session

Commands: [**4_digit_message_code**],**PathCntl**,[textstring]
Responses: [**4_digit_message_code**],**PathCntl,[ACK**|**NACK**],[textstring]

**0000**,**PathCntl**
First command after ssh session as been established, i.e. authentication and authorization checked and cipher changed to NONE cipher.

**0000**,**PathCntl,ACK**
Positive answer of server to the 0000 control session initiation cmd.

**0001**,**PathCntl,NOOP**
Command sent by client to check if server is still available.

**0001**,**PathCntl,NOOP,ACK**
Acknowledgement to a previous sent 0001 message by client process.

**0002**,**PathCntl,NOOP**
Command sent by server to check if client is still available.

**0002**,**PathCntl,NOOP,ACK**
Acknowledgement to a previous sent 0002 message by server process.

**8000**,**PathCntl**
This command is sent by client process for finalizing control connection.

**8000**,**PathCntl,ACK**
Positive response of server to a finalizing request 9000 by client.

**9000**,**PathCntl**
Client or server kill control connection because of severe problem imediately.

# FiTP commands & replies
# Grant access (1)

Commands: [**4_digit_message_code**],**GAcR**,[textstring]
Responses: [**4_digit_message_code**],**GAcR**,[**ACK**|**NACK**],[textstring]

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

Client 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 quadrupels 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).

Allows easy scanning by firewall hardware (positional parameters, which can be scaned by 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).

This allows the server to send responses related to each individual grant request by specifying this decimal number

# FiTP commands & replies
# Grant access (2)

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

*Positive acknowledgement to the grant request n. parms see above.*

**3001**,**GAcR,NACK,Form_error**

Server 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.

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

Positive feedback from server. Access grants will be destroyed.

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

Server waits for 3000 or 3020, but got malformed or different message.

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

Client has requested to delete an access grant, not known to server.

# FiTP commands & replies Grant access (3)

**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

# FiTP commands & replies Structure of codes (1)

There are five values for the first digit of the comand and reply code:

**0xyz**
Control session startup

**1xyz**
Key exchange phase

**2xyz**
Authentication and authorization phase in progress.

**3xyz**
Grant access commands and replies are being sent.

**8xyz**
Control connection in closing process.

**9xyz**
Client &/or server has diagnosed a severe problem.
Control connection will be killed immediately.

# FiTP commands & replies
# Structure of codes (2)

Digit two (x) is phase dependent.
e.g. phase 3 grant access commands x = 4 (IPv4) and x= 6 (IPv6)

Digit three (y) corresponds to different states communication has reached

Digit four (z) Response messages are defined as follows:

**xyz0**
Indicates a positive response. Server received the request correctly and has changed its state accordingly.
For grant access requests server acknowledges that the request has been accepted.

**xyz1**
Server received a message from client which is illegally formatted, out of sequence or corrupted.

**xyz**{n}
{n ϵ 2, … , 8}: negative response. Previous request cannot be acknowledged.

**xyz9**
Negative response. Previous request cannot be granted.
Dependent on phase, it has to be started again or 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.)

# Command Syntax

**Commands are text strings beginning 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.**

**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***

**Argument field: Variable length character string ending with ASCII <CRLF>.**

**Any text string following x”00” will be ignored and deleted.**

**The syntax is specified in 8bit-ASCII.**

# FiTP commands
# Option syntax (1)

**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 >

# FiTP command Option syntax (2)

<five-digit-integer> ::= < “00001” | “00002” | … | “65535” >

*i.e. any integer [1..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..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”>

<v6netprefix> ::= < “000” | “001” | … | “128” >

<port> ::= <five-digit-integer> | <”65536”> *i.e. any decimal integer [1..65536]*

# Security Considerations (1)

FiTP allows easy dynamic configuration of firewall systems by authorized users.

Comparison to IPsec communication:

Security is assumed to be handled by the receiving server process.

Authentication and authorization has been checked and we trust data connection.

If receiving side allows packet forwarding to internal hosts, tunneling can be done. So with IPSEC we rely on security check by the server system.

Same principle for FiTP:

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 our organization normally protected by our organizational firewall.

Again internal server has checked authentication and authorization.

Main difference:

With FiTP we know, where communication streams are going.

# Security Considerations (2)

So can we trust the FiTP protocol itself?

FiTPl uses common techniques for secure communication.

It uses well known ssh protocol for encapsulation of FiTP commands.

Common authentication and authorization techniques to check if the remote user is allowed to request port openings via SSH.

Authentication and authorization process is encrypted.

No man-in-the-middle can intercept and modify messages exchanged because of HMAC routines which protect grant requests grants

Any message sent 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.

# Summary

- The FiTP protocol described 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 periods though they are used only within short time period.
- Constantly open ports are a potential security risk which can be minimized when configured only in time periods where really needed.
- Usage of FiTP within grid applications minimizes the time period in which traffic can pass the firewall.
- FiTP allows opening of ports without manual interaction of a firewall administrator.
- Leads to fast configuration independent of administrator availability.
- Authenticated and authorized interaction between user applications and authentication/authorization servers provides a secure configuration of the firewalls involved.

# Questions and discussion