Addressing the Tension Between Strong Perimeter Control and Usability

Thomas H. Hinke
NASA
Ames Research Center MS 258-5
Moffett Field, CA USA 94035-1000
1-650-604-3662
Thomas.H.Hinke@nasa.gov

Paul Z. Kolano
AMTI
Ames Research Center MS258-6
Moffett Field, CA USA 94035-1000
1-650-604-4271
kolano@nas.nasa.gov

Chris Keller
AMTI
Ames Research Center MS258-6
Moffett Field, CA USA 94035-1000
1-650-604-5597
ckeller@nas.nasa.gov

ABSTRACT

This paper describes a strong perimeter control system for a general purpose processing system, with the perimeter control system taking significant steps to address usability issues, thus mitigating the tension between strong perimeter protection and usability. A secure front end enforces two-factor authentication for all interactive access to an enclave that contains a large supercomputer and various associated systems, with each requiring their own authentication. Usability is addressed through a design in which the user has to perform two-factor authentication at the secure front end in order to gain access to the enclave, while an agent transparently performs public key authentication as needed to authenticate to specific systems within the enclave. The paper then describes a proxy system that allows users to transfer files into the enclave under script control, when the user is not present to perform two-factor authentication. This uses a pre-authorization approach based on public key technology, which is still strongly tied to both two-factor authentication and strict control over where files can be transferred on the target system. Finally the paper describes an approach to support network applications and systems such as grids or parallel file transfer protocols that require the use of many ports through the perimeter. The paper describes a least privilege approach that dynamically opens ports on a host-specific, if-authorized, as-needed, just-in-time basis.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection – *Access controls, authentication.*

General Terms

Design and Security.

Keywords

Two-factor authentication, security-based dynamic port control.

1. INTRODUCTION

This paper describes a multifaceted system that provides strong perimeter control for a large, general purpose NASA processing systems, while taking significant steps to address usability issues, thus mitigating the tension between strong perimeter protection and usability.

The system described in this paper was the result of a design and implementation effort to provide an improved perimeter protection system for a newly acquired NASA supercomputer – the Columbia system [3]. As will be shown in the paper, significant steps were taken to ensure that usability did not suffer as the strength of the perimeter protection system was raised over that used with previous NASA supercomputers.

The next section of this paper will describe the overall processing environment that is protected by the perimeter control system. Then section 3 provides a brief description of the threats that the system is designed to counter. Section 4 provides a description of the security policy that is enforced. Section 5 describes the overall security architecture and the following three sections describe the three different sets of components that comprise the perimeter protection system. The final sections describe related work, conclusions and acknowledgements.

2. PROCESSING ENVIRONMENT

The Columbia system is a 10,240 processor system that consists of a set of 20 backend nodes, with each node consisting of 512 processors. A 64 processor Columbia front-end system provides interactive support for job preparation and job submission to the backend nodes.

The system supports unclassified, scientific and engineering processing for all of NASA's mission directorates: Aeronautics Research, Science, Space Operations and Exploration Systems. Each of the NASA mission directorates has a percentage of the Columbia system and they are each responsible for determining how users will be selected to use the assigned percentage. Some directorates use a general call for proposals while others will have particular problems that need to be addressed and will assign an allocation to those who are working on the problem.

While some users are located at the Columbia facility, most users are geographically distributed, coming from other NASA centers as well as various companies and universities. For the most part, these users develop their own programs and execute them on Columbia in order to support their scientific research or engineering analysis. Programs are not pre-vetted prior to being allowed to run on the Columbia system, hence the Columbia system must be considered a wide open system that may need to support any possible type of program that may be needed to support scientific or engineering processing.

In addition to having to support all manner of programs, the Columbia system also must be accessible from the Internet, since users are geographically distributed. In addition, to support the large data transfer requirements of many users, Columbia will also

be connected to the National LambdaRail [10], a 10 Gigabit/sec national, high-speed optical network.

While some interactive processing is allowed, for the most part, access to Columbia is through a batch scheduling system, which is the Portable Batch System (PBS). One of the implications of running batch jobs is that the user is not necessarily present at the time that the job is run or at the time that the job needs to transfer some data. In contrast, most access to the Columbia front-end is interactive in order to perform job preparation, compilation and job submission.

While the Columbia system is the main computation system, it works closely with a number of other systems, such as a mass storage system, a high-end visualization system as well as other much smaller general purpose processing systems that are co-located within the same facility.

A processing factor that had a significant impact upon the design of a perimeter control system was the desire to allow Columbia to support network protocols or systems that are characterized by needing to open a large number of TCP/UDP ports. For the software in question, the protocol determines the specific ports that need to be opened at run time, from a large port address space. Currently Columbia supports the parallel file transfer protocol bbFTP, which opens a number of ports in parallel to achieve faster transfer rates.

Another network system that has also been considered for future support, is grid computing such as that provided by the Globus system [5] and which is being standardized by the Global Grid Forum [6]. This system uses a large number of ports selected from a large port address space. One researcher who has looked at this issue from the perspective of access through a firewall reports that "The firewall port-opening approach favored by the Globus Project has been shown to work, after some minor adjustments, with a default Globus deployment. However, an outstanding issue of illegal port usage (not adhering to port range restrictions), for commands that create a return connection request, still remains. While testing was successful at the 3000 to 6000 port range, it is clear that such a small fixed range is too restricted to be of much value in a production grid environment." [2] What this means is that allowing 3000 ports to be accessed in the range from port 3000 to port 6000 is viewed as too restrictive for a production system, meaning that production systems may need to allow even a larger range of ports to be utilized.

The need to open a significant number of ports for the Condor grid system is also recognized, although in this case the port range is restricted to approximately 255 ports [9]. The point is that grid-type software requires the opening of a significant number of ports, which opens up vulnerabilities for attacks on the services that may be using these ports.

While currently not grid enabled, the security design anticipated possible future grid access to Columbia and thus had to ensure that nothing was done that would preclude the Columbia system's support for grid processing such as that provided by the Globus suite of tools [5], or other network applications that may need many open ports.

In summary, the Columbia system must be able to support the following:

- Must be accessible from the Internet
- Must be accessible at up to a 10 Gigabits/sec rate
- Must support the execution of user developed code with no pre-vetting
- Must support file transfers initiated by batch jobs at a time when the user may not be currently logged onto the system
- Must support network systems that dynamically open a large number of ports

The goal in developing a security system for Columbia was not to hinder the ability of users to do a wide range of different types of processing, while ensuring that the system will not be left open to the threats that will be considered in the next section.

3. THREATS

The perimeter control system technology described in this paper is directed at countering external threats from non-users. In this case, external threats emanating from people and computers who are located anywhere on the Internet as well as people and computers that are located within the NASA division that houses the Columbia system. The perimeter control system mediates access by all users of the Columbia system, irrespective of their location.

The types of threats that are addressed by the perimeter protection system are all of those that can be directed by an external entity against the system. This includes the man-in-the-middle attack in which a malicious entity is assumed to be monitoring and perhaps modifying the data that passes into and out of the system under attack. The man-in-the-middle attack can, for example, be used to capture passwords. Threats of concern also include externally mounted attempts to directly attack exposed services in order to gain unauthorized access to the system or mount a denial-of-service attack. As has been noted, since the Columbia system is accessible from the Internet, it is open to attack from virtually any Internet connected computer.

Since all Columbia users are highly vetted, there is less concern with attacks mounted by legitimate users. Techniques used to address these potential threats are beyond the scope of this paper.

4. SECURITY POLICY

The security policy for the Columbia system is rather simple and includes the following two properties:

1. *Outside Initiated Policy:* Access initiated from outside of the Columbia system must be controlled to ensure that only legitimate users can access Columbia and its associated systems.
2. Inside Initiated Policy: Access initiated from Columbia and its associated systems is unrestricted .unless there is some security reason to deny connections to certain site, but it is anticipated that there will be very limited exceptions to this policy.

The *inside initiated policy* is required since the Columbia users have a wide range of different types of processing that needs to be supported and this support may require them to connect to a wide variety of different systems at different locations. The users' needs in this area are much too fluid to enforce any security constraints for access initiated from Columbia and its associated systems.

5. SYSTEM SECURITY ARCHITECTURE

One of the guiding principles in designing the security architecture for Columbia was that it should satisfy the requirements of a security reference monitor [1]. A security

reference monitor is an abstract concept for a security mechanism that mediates access between active subjects, such as processes, and passive objects, such as files or remote machines. The requirements of a security reference monitor are that it:

- Be tamper proof, so that it can not be modified by malicious code
- Be always invoked, so that it can support complete mediation
- Correctly enforce the desired security policy

For the Columbia system, the security reference monitor mediation is distributed over several components that collectively enforce the desired security policy. The systems that comprise the overall Columbia reference monitor include the following:

- Reference monitor for interactive access
- Reference monitor for unattended file transfers
- Reference monitor for network access

Interactive access is any type of access where the user is present to perform authentication, which for the Columbia system is two-factor authentication involving a physical authentication fob. Since much of the Columbia workload involves batch jobs that may run at times when the user is not present to perform authentication, the unattended file transfer reference monitor was identified as a needed component to couple authentication associated with file transfers to two-factor authentication, even though the user was not present (at the time of the transfer) to perform this authentication. The final reference monitor of the triad is concerned with mediating network access in order to reduce the number of ports that have to be opened statically.

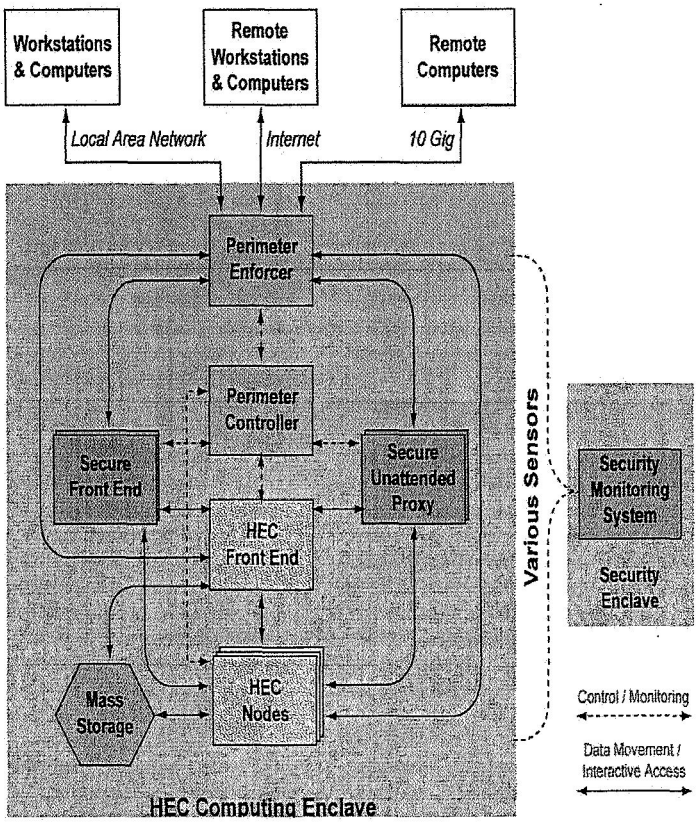

Figure 1: Enclave Security Architecture

Figure 1 illustrates the overall security architecture of the system. The Columbia system and all of its associated processors are placed inside of a high-end computing (HEC) protected enclave. All users, including those who are co-located with the Columbia system as well as remote users, are considered to be outside of the enclave, and therefore are subjected to identical access controls. The various reference monitors that comprise the perimeter control system have the task of mediating access between external subjects, represented by users at various local and remote computers, and the Columbia system and its associated processors that are located within the HEC enclave. The components that implement the three reference monitors for the enclave are as follows:

- Secure Front End for interactive access, which performs two-factor authentication.
- Secure Unattended Proxy, which supports file transfers when the user is not present to perform two-factor authentication
- Perimeter Enforcer/Controller, which dynamically opens and closes ports

Also shown in the figure is the security monitoring system, which is shown for completeness, but will not be discussed since it is beyond the scope of this paper.

The next three sections will describe each of the component sets that comprise the enclave's reference monitors.

6. INTERACTIVE ACCESS CONTROL

The Secure Front End (SFE) enforces two security requirements:

- All interactive access to the SFE and systems within the enclave must use an encrypted communications channel, with the SSH protocol providing the encryption.
- All interactive access to systems within the enclave must undergo two-factor authentication, with RSA's SecurID [11] being the authentication mechanism used.

The encryption provided by SSH ensures that authentication information as well as other information sent from the user's system to the SFE is protected as it flows between systems. Of course, if users are accessing the enclave through a number of SSH hops through different systems, the information being communicated will be in the clear in these intermediate systems and thus subject to compromise through a man-in-the-middle attack. To prevent this, users should log directly into the SFE from their home system. To ensure the protection of information flowing between the SFE and systems within the enclave, SSH is also used from the SFE to the enclave systems. SSH provides both remote terminal-type access as well as file transfer access using the scp protocol.

The use of SecurID provides a means to counter vulnerabilities that exist since Columbia users come from systems with varying levels of security. It addresses cases in which a user's Columbia password is captured on the remote system and also the case where a user might use the same password on Columbia as on a compromised remote system. SecurID authentication uses a physical device or fob that displays a pseudo-random number that changes every 30 seconds. As part of the authentication processes, the user must enter the current value displayed on the fob.

As a physical device, the fob represents something that the user has, which is the first factor in the two-factor authentication. SecurID authentication also requires that the user memorize a personal identification number (PIN), which provides the second factor in two-factor authentication. The fob and the associated number displayed are unique to each user.

To log into the SFE, the user must present his PIN and the current pseudo-random number that is displayed on his fob. The SFE then accesses the SecurID server that supports the enclave to determine if the PIN and pseudo-random number provided by the user match the PIN and current pseudo random number known by the server. If there is a match, then the user is successfully authenticated.

The regulations that mandate the security requirements for the Columbia system require a password that includes three of the four types of characters (upper and lower case letter, number and punctuation). Unfortunately, SecurID with its time-varying pseudo-random numbers and its PIN supports only two types of characters, numbers and letters, with no distinction made for case. Thus, the SFE must also enforce password or public key authentication as well as SecurID.

In addition, after authenticating to the SFE, users have to again authenticate to the particular system within the enclave that they want to use, since each system must support only authenticated users and not all users have privileges to use all of the systems located within the enclave.

6.1 Improved Usability

With the use of two-factor authentication and the need for all users to authenticate at the SFE and also at the particular system within the enclave that they want to use, a tension existed between strong authentication and ease of use. To provide some mitigation for this tension, the SFE was designed with a capability that we call SSH-pass-through. SSH-pass-through involves a combination of ssh-agent and public key authentication. To use SSH-pass-through, a user must place his public key on both the SFE and on the systems within the enclave that he wants to use. He must also provide the following in the ssh config file on his local system:

```
Host columbia-sfe1
  ProxyCommand ssh userA@sfe1.xxx.yyy.zzz \
    /usr/local/bin/ssh-proxy columbia.xxx.yyy.zzz
```

In the actual configuration file, xxx.yyy.zzz represent the remainder of the address for sfe1 and Columbia, which in this example are assumed to be on the name subnet.

When a user issues the command ssh columbia-sfe1, this will initiate a connection to the IP address sfe1.xxx.yyy.zzz. Since all access to the SFE requires two-factor authentication, the user will be prompted to perform his SecurID authentication. The SSH daemon will also perform public key authentication with the ssh-agent on the user's system.

Note that the user had to perform only the SecurID authentication – the public key authentication was invisible to him since it was performed by his ssh-agent running on his local workstation. The configuration file then indicates that the ssh-proxy program on sfe1 should be invoked using IP address columbia.xxx.yyy.zzz as an argument. This will establish an ssh connection with the ssh daemon on the Columbia system, which will perform public key authentication with the ssh-agent on the user's local workstation. Again, this is invisible to the user, since it will be performed automatically by his ssh-agent. The net result of this is that the user personally had to perform only SecurID authentication with the SFE, with public key authentication at the SFE and again at the Columbia system being performed automatically on his behalf by his ssh-agent. Through this simple approach, the tension between providing a higher level of authentication, while still facilitating usability was reduced.

6.2 Implementation Issues

Because the SFE represents a single point of failure for access to the systems within the Columbia system enclave, two identical SFEs are provided. A significant design issue that had to be addressed was the nature of the interaction between the two SFEs and whether the SFEs should project a single SFE at a single IP address, or two SFEs, each at their own IP addresses. Various approaches were considered for supporting the single IP address, but they either involved additional complexity to deal with failure detection and hand-off, or they complicated the task of managing the public keys required for SSH pass-through. Having separate, independent SFEs simplified the design, since neither had to be aware of the state of the other. In general, for security systems

with equivalent capability, the best choice is usually to select the simpler design approach.

In support of the *tamper proof* requirement of a security reference monitor, the SFE has been developing using a Linux operating system that was stripped of all but the functional capability needed to support the SFE's identification and authentication functions. Since users will actually log onto the SFE to install their public keys (for SSH pass-through), the SFE was developed with a strong effort to minimize the amount of functional capability that was installed on the system and also limit the functions that are available to users. For example, users do not even have the ability to make a directory, since this is not a function that they need to either log through the SFE or transfer a file using scp. All users are restricted to a very limited execution environment on the SFE.

This *tamper proof* requirement is further supported by the physical separation between the SFE and the other systems within the enclave. This means that changes to the computers within the enclave do not affect the SFE software. This physical separation also means that the SFE is physically isolated from any malicious processing that might take place on the Columbia system's front end or backend nodes. The reduced functionality and physical separation both support, but do not guarantee, the *tamper proof* requirement of a security reference monitor.

Another advantage of the SFE approach is that the SFE represents a single point for imposing two-factor authentication for users, rather than having each of the systems within the enclave having to support this.

Each SFE is implemented on dual hyper-threaded 3.2 Gigahertz processors with a Gigabyte of RAM running the Linux operating system. The two identical SFEs, provided for redundancy, are each capable of handling the whole SFE load. Of note is the fact that this is just a commodity system that has no particularly unique requirements and is available for a modest cost.

7. UNATTENDED ACCESS CONTROL

A major challenge in the development of the perimeter protection system for the Columbia system was support for unattended file transfers within an environment that required the use of SecurID. As has been noted, SecurID authentication requires that a user read a time-varying pseudo-random number from a physical SecurID fob device and then enter it and a PIN in order to authenticate. The problem arises if a file needs to be transferred into the enclave as part of a script or other program running on a remote system at a time when the user is not present with his fob to perform the SecurID authentication. This represents yet another tension between security and usability.

The solution to this problem is the Secure Unattended Proxy, which is a security component that allows a user to preauthorize a number of unattended file transfers for a limited future period using his SecurID fob. This authorization is based on the user's acquisition of a limited duration public/private key pair, which is obtained based on SecurID authentication. In this way, SecurID authentication is tightly bound to the unattended file transfer, since the key pair needed to perform the transfer required the use of SecurID to obtain it. In addition, the actual file transfer requires the use of public key authentication, which is much more resistant to compromise than are passwords.

7.1 Use Scenario for Improved Usability

Under this approach, the user accesses a key server and performs identification and authentication with SecurID and a password. This triggers the creation of a public/private key pair. The key server deploys the public key to the Secure Unattended Proxy component and to the target machine, such as the Columbia system. The user is then provided with a private key, which he can save on his system. This key is good for a limited period of time, such as a week. After this time, the Secure Unattended Proxy will no longer accept that key for authentication and the user must generate another.

At the time of the file transfer, the private key must be available to the software, such as a script, that is performing the transfer. The script initiates the file transfer through the Secure Unattended Proxy. The Secure Unattended Proxy verifies that the file transfer command is one of the authorized protocols (e.g., scp, sftp or bbFTP), and that the arguments that accompany the command are consistent with security concerns. If necessary the Secure Unattended Proxy will re-write the command such that it complies with the security policy. This rewriting is very critical to the security of the system, hence it must be carefully validated for correctness. To use the system, the private key must be loaded into an ssh-agent, after which scp (or sftp or bbFTP) transfers can be performed using an appropriate command such a the following which is used for an scp transfer:

```
scp -S scpwrap file target.xxx.yyy.zzz:newfile
```

where "scpwrap" is a wrapper that initiates the transfer through the Secure Unattended Proxy:

```
#!/bin/sh
exec ssh sup1.xxx.yyy.zzz
```

For bbFTP, transfers can be performed using a command such as:

```
bbftp -L "ssh -q sup1.xxx.yyy.zzz ssh -q" \
-e "put /dir/file newfile" target.xxx.yyy.zzz
```

Once the Secure Unattended Proxy is satisfied that the file transfer command complies with the security policy, it forwards the command on to a dedicated SSH daemon on the target system, such as the Columbia system front end. Security code associated with the daemon ensures that the Columbia system actually invokes the command string that has been provided. For a protocol such as bbFTP, the SSH daemon will spawn a bbFTP server that will create new connections between the Columbia system and the remote site that has requested the file transfers. These connections are dynamically authorized to pass through the perimeter using the perimeter controller, as will be described in section 8.

7.2 Vulnerabilities and Countermeasures

The temporary private key file cannot be encrypted or else the unattended batch processes cannot use it. As a means to impose security restrictions on unattended operation, the directories into which files can be transferred are limited when using the Secure Unattended Proxy. If the user's private key were to be compromised, this would prevent the unrestricted overwriting of files within the use's directory space.

Security relevant files such as the user's public key can never be written. For other files, the user must specify the set of directories to which the unattended file transfer may write, using a configuration file in the user's home directory. Creation of this configuration file must be done through a separate interactive session, which requires SecurID. This configuration file specifies which directories can receive the unattended file transfers. No transfer will occur if at least one directory is not listed in the configuration file. This is enforced by dynamically injected code in the SSH server of the target machine.

While it is true that the one week key lifetime does introduce a one week window of opportunity for an intruder to compromise the system, under the current design, even if a potential intruder were to capture the public key, the damage that could be caused would be carefully limited to being able to dump files to only to the preauthorized directories.

While work is underway to increase the capability of this unattended access, an attempt will be made to continue to limit the damage that could be caused by the capture of a private key.

8. NETWORK ACCESS CONTROL

One of the significant accomplishments of the Columbia security project has been the implementation of an approach for supporting network-oriented software that requires a relatively large number of ports to be opened through the perimeter protection system. Such software includes grid software as well as high performance file transfer software such as bbFTP. The goal is to address the tension between supporting network-oriented software that wants to open up a significant number of ports, without allowing many holes (ports) to be opened through the perimeter in case these network systems should need a particular port.

The Perimeter Enforcer and the Perimeter Controller, introduced in the architecture section, are the two components that provide a capability to dynamically open and close ports in response to the needs of network-oriented software operating within the Columbia system enclave.

In order the be able to support network traffic at the leading edge of commercial technology (currently 10Gigabits/sec), the Perimeter Enforcer is a network device, which controls access by dynamically changing access control lists (ACLs). The Perimeter Controller is a commodity computer that interacts with software on the systems within the enclave to determine what ports need to be opened, and then commands the Perimeter Enforcer to open and when appropriate, close the specified ports.

As has been noted, the security policy for the Columbia system enclave is to permit all internally initiated connections, unless there is some security reason to deny connections to certain sites, but it is anticipated that there will be very limited exceptions to this policy. In addition, all incoming connections that are in response to an internally initiated connection are also allowed, if the relationship of this incoming connection to an internally initiated connection can be accurately established.

The challenge comes with externally initiated connections that are not in response to an internally initiated connection for which the relationship can be established. In this case, the default setting for the Perimeter Enforcer is to deny all incoming connections.

The goal of the Perimeter Enforcer is to dynamically open port access through the perimeter, on a host-specific, if-authorized, as-needed, just-in-time, dynamic basis. Experiments have shown that this can be accomplished by instrumenting the application so that this application instrumentation can request the Perimeter Controller to open or close specified ports through the perimeter. The Perimeter Controller then takes the information from all of the applications and compiles this into changes in the ACLs on the Perimeter Enforcer.

Because of the need to support 10 Gigabits/second data network traffic, at this point in time the only devices that can handle this high data rate are network devices such as routers or switches – firewalls are not currently capable of handling these high data rates. Also, in general, the speed of firewalls will likely always lag behind the speed of network routers and switches. Hence for systems such as the Columbia system, which will always be operating on the frontier of speed, a security approach that relies on firewall technology will probably never be appropriate since it will always lag behind routers and switches.

At this point, there may be some confusion as to how the role of such systems as the SFE or the Secure Unattended Proxy (which each support file transfers) relates to the role of the Perimeter Enforcer/Controller, whose purpose is to open ports for network software such as for file transfers. For protocols such as SCP, if the transfer is initiated through the SFE, then both the control connections and the data connections travel through the SFE. The Perimeter Controller allows unrestricted access to the SSH daemon ports on the SFE as well as the appropriate ports on the Secure Unattended Proxy. For a high performance transfer such as one involving bbFTP, while the initial control channel would travel through the SFE (for a user initiated interactive transfer) or the Secure Unattended Proxy (for a script initiated transfer), the resulting data channels would not be mediated by the SFE or Secure Unattended Proxy. The data channel would, however, be mediated by the Perimeter Enforcer. Ports would be dynamically opened in response to commands from the Perimeter Controller, based on instrumentation of the bbFTP software.

9. RELATED WORK

As one means of assessing the state of related work, at its inception, the Columbia system project performed site visits to six of the nation's major high performance computer centers and performed an interview with a seventh. The observed architectures from these visits were used to formulate a taxonomy that is shown in Figure 2. Note that this taxonomy relates to just the seven sites that were visited or interviewed. The basic constructs of this taxonomy could be extended to other architectures, but this is not the objective of this analysis, since we wanted to understand what other high-end computer centers were actually doing.

At the top of the figure is the fully open type, which imposes no additional security controls on access to the high end computer other than those provided by its own identification and authentication system. At the other extreme (at the bottom of the figure) is the virtual private network (VPN) type, which establishes an encrypted link between the user's client computer and the high-end system. In this particular case, mirroring one of the visited sites, this types also includes router enforced access restrictions based on static access control lists (ACLs). Because VPN supports an encrypted link from client to high-end system, special VPN client software would have to be installed on all client machines, which we did not want to impose on the Columbia system users.

Architecture types that impose some mediation (other than encryption) on external access to the high-end system represent

the middle ground of the systems visited or interviewed. In the static ACL type, ACLs on a network router are used to restrict which external sites can communicate with the high-end system. A slight variation of this type, which moves toward the dynamic type is an approach in which external sites are automatically

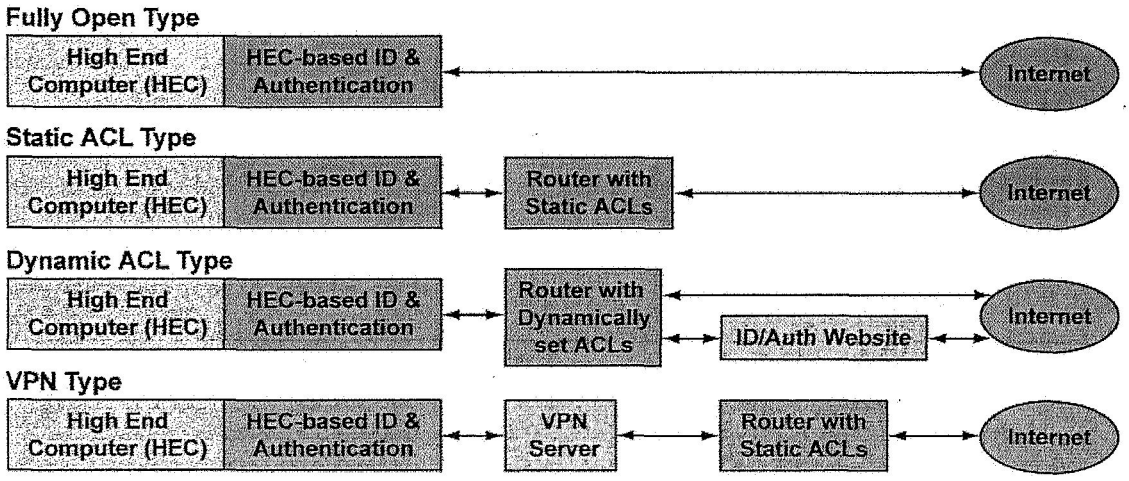

Figure 2: Taxonomies of Security Architectures

blocked using ACLs if these external sites are caught scanning the high-end computer. In this case, the block is dropped after a certain amount of time has elapsed with no additional scans.

In the dynamic ACL case, this mediation is dynamically configured in response to user requests, with the default being no access by external users. In the particular type shown in the figure, an identification/authentication web site is used for user identification and authentication and then when successful, an ACL is dynamically set on the router to allow the IP address associated with the successfully authenticated user to establish connections to the high end system through the router.

The Columbia system expands on this by using the Perimeter Enforcer/Controller to authorize a finer granularity of perimeter access (port) rather than the IP address. A key server authorization approach is used in the Secure Unattended Proxy, but an in-line SFE approach was selected for the Columbia system.

The concept of a secure front end is not new, going back to the very early days of the computer security community. Some early references to this idea can be found in the SDC communications kernel work [7] and the communications operating system network front end COS/NFE work [8]. While these were both high assurance projects, the idea of a front-end processor to securely handle functions for a main computer go back to the early days of the field.

A form of dynamic port control is provided by stateful firewalls such as [3], which interpret specific protocols to determine when a given port needs to be opened between a given pair of hosts. For example, in the case of FTP, these firewalls will allow only data connections to an FTP server from hosts that have successfully authenticated to its control port. Such support is generally limited to a small set of standardized protocols, however, thus cannot be applied to new or emerging protocols such as those for the grid. Protocols that use encryption over the control channel also cannot be supported. Finally, current high-end stateful firewalls do not yet support inspection at the full 10 Gibabit/sec line-rates already supported by core routers and switches, thus are not suitable in high performance environments.

Several projects exist for limiting the user to specific actions when connecting via ssh. For example, rssh [12] and scponly [14] can be used to allow only file transfers through authenticated ssh sessions. Both of these projects, however, require changes to user shells on the target system, which makes it impossible for those users to access the system for other purposes even when logging in interactively. The Secure Unattended Proxy enforces a similar policy for unattended transfers, but does not affect user shells on the target systems, thus they can still log in interactively with full functionality. It also allows a finer granularity of control over which files can be written on the target system.

10. CONCLUSIONS

The Columbia perimeter control system has provided a high level of perimeter protection for general purpose processing systems in general and a large supercomputer in particular, while mitigating the tension that can result in ensuring a reasonable level of usability. As has been shown in the paper, SFE authentication is strong with the two-factor authentication provided by SecurID, but usability is enhanced with the SSH-pass-through capability. While all processing on Columbia must be tied to SecurID authentication, the tension that this requirement has with unattended, script-based file transfers has been mitigated though the use of SecurID to pre-authorize a limited duration, limited capability for future file transfers. The tension between the need for network-oriented software to open many ports in the course of performing its function and the need for the perimeter to lock-down the number of ports that are kept open has been mitigated by an approach which dynamically opens ports on an a host-specific, if-authorized, as-needed, just-in-time, dynamic basis. All of this is accomplished automatically, without the user's knowledge.

In the course of mitigating this tension between strong perimeter protection and usability, Columbia's perimeter security design has also satisfied many computer security principles [13].

Dynamically opening and closing ports so that they are available only when needed and closed at other times is an example of the security concept of least privilege. The overall architectural design is based on the complete mediation provided by a security reference monitor. The physical separation of the various security components that provide the perimeter's security reference monitor supports the reference monitor's tamper proof property. Minimizing the capabilities available to the SFE users also seeks to eliminate capabilities that the users could use to gain unauthorized access, which also supports the tamper proof requirement. The use of front-end systems such as the SFE for authentication while still requiring authentication on the particular system to be accessed within the enclave is an example of defense in depth. The SUP exhibits separation of privilege since an interactive session through the SFE is required to set up the configuration file to authorize transfers into particular directories, while the SUP itself mediates the actual transfer.

Even though Columbia is an unclassified system, the overall design has attempted to provide a high level of protection, while still ensuring usability. Although the techniques have been applied to a supercomputer, we believe that they are equally applicable to other systems that support general purpose processing from a large number of distributed users

11. ACKNOWLEDGMENTS

We would like to acknowledge the contributions of Derek Shaw, who developed the underlying software for the SFE and integrated it to SecurID. We would also like to recognize Todd Welch and to Dave Tweten for their contributions to the project.

12. REFERENCES

[1] Anderson, J.P. 1972. *Computer Security Technology Planning Study*, ESD-TR-73-51, Vol. 1, Hanscom AFB, MA. 1972.

[2] Baker, M, Ong, H. and Smith, G. *A Report on Experiences Operating the Globus Toolkit Through a Firewall.* Sept. 2001. (Available at http://dsg.port.ac.uk/projects/grid-security/docs/globus-firewall-experiences.pdf)

[3] Cisco Systems, Inc.: *Benefits and Limitations of Context-Based Access Control*, Jul. 2004. (Available at http://www.cisco.com/warp/public/110/36.html)

[4] Columbia System, http://www.nas.nasa.gov.

[5] Foster, Ian, Kesselman, Carl, and Tuecke, Steve. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3).

[6] Global Grid Forum, www.ggf.org.

[7] Golber, D.L. A Practical Executive for Secure Communications. *Proceedings of the Fourth Seminar on the DoD Computer Security Initiative*, National Bureau of Standards, Gaithersburg, MD, August 1981.

[8] Grossman, Gary. A Practical Executive for Secure Communications. *Proceedings of the 1982 Symposium on Security and Privacy*, Oakland, 1982.

[9] Kewley, J. *Using Condor Effectively in the Presence of Personal Firewalls*. Oct 2004. Available at http://tardis.dl.ac.uk/Condor/docs/FW_condor.pdf.

[10] National LambdaRail, http://www.nrl.net

[11] *RSA SecurID Authenticators*, RSA Corporation, Available at http://www.rsasecurity.com/node.asp?id=1156

[12] rssh. Available at http://www.pizzashack.org/rssh.

[13] Saltzer, J.H., Schroeder, M.D.: The Protection of Information in Computer Systems. *Proceedings of the IEEE*, vol. 63, num. 9, 1975.

[14] scponly. Available at http://www.sublimation.org/scponly.