An Inter-Realm, Cyber-Security Infrastructure for Virtual Supercomputing

Jalal Al-Muhtadi[*∇] Wu-chun Feng[∇] Mike Fisk[∇]

[*] *Department of Computer Science,*
University of Illinois at Urbana-Champaign
Urbana, IL 61801
almuhtad@uiuc.edu

[∇] *Research & Development In Advanced Network Technologies (RADIANT)*
Computer and Computational Sciences Division
Los Alamos National Laboratory
Los Alamos, NM 87545
{feng,mfisk}@lanl.gov

ABSTRACT

Virtual supercomputing, (i.e., high-performance grid computing), is poised to revolutionize the way we think about and use computing. However, the security of the links interconnecting the nodes within such an environment will be its Achilles heel, particularly when secure communication is required to tunnel through heterogeneous domains. In this paper we examine existing security mechanisms, show their inadequacy, and design a comprehensive cyber-security infrastructure that meets the security requirements of virtual supercomputing.

Keywords

Security, virtual supercomputing, grid computing, high-performance computing, GSS-API, SSL, IPsec, component-based software, dynamic reconfiguration.

1. Introduction

The rapid growth in high-speed networks and the ubiquity of computers have converged to enhance global interconnectivity and provide the foundation for a new kind of computing infrastructure – computational grids [9]. These grids consist of computational nodes that are distributed throughout geographically dispersed areas and interconnected through a reliable internetwork; they harvest significant processing power, memory, and resources by utilizing the capabilities offered at the end nodes.

When several high-performance computing nodes (e.g., Cray SV2 and SGI Origin 3000) or very large numbers of "ordinary" computing nodes (e.g., all the PCs at Intel [11]) are aggregated together in a grid, the grid is oftentimes referred to as a virtual supercomputer which can rival or outperform conventional supercomputers. Hence, virtual supercomputing is poised to revolutionize the way we think about and use computing.

However, the security of the links interconnecting the supercomputing nodes will be the system's Achilles heel. To fully reap the benefits offered by such a system, we need a cyber-security infrastructure that can provide highly customizable and dynamically reconfigurable services to secure communications between the nodes of a virtual supercomputer. Because widespread deployment of virtual supercomputers involves using the Internet as the communication backbone, the cyber-security infrastructure must seamlessly deal with the heterogeneous nature of the Internet.

To design such a cyber-security infrastructure, we must understand the obstacles presented by the Internet. The Internet is divided into different areas each having diverse properties and characteristics, such as different link-layer technologies, media types, security requirements, bandwidth capacities and so on. Areas with similar network characteristics, e.g., a campus network, are referred to as *realms* [8]. To handle this diversity, special devices are introduced at the realm boundaries. These devices transparently fix packet flows between endpoints, handle data transition between realms, and provide important additional functionality. The added functionality includes (but is not limited to) mobility support, address translation, packet filtering, data confidentiality and authentication, data compression, and special processing to improve performance over particular realms. These special devices are referred to as middleboxes. Examples of middleboxes include firewalls, proxies, network address translators, active wardens, and wireless gateways. However, these middleboxes are harmful as they often violate the end-to-end nature of conventional Internet applications and hinder the operation of existing and new end-to-end protocols [8]. One common

example is the Network Address Translation protocol (NAT) [6] [22]. NAT allows a private network of devices with non-global IP addresses to connect to the Internet by sharing a limited number of global IP addresses. The NAT gateways translate between internal IP addresses and corresponding global addresses. This simple address translation breaks many upper-layer protocols. For example, the control connection for `ftp` transmits the IP address and TCP port to use for the data connection which is made in the opposite direction [20]; therefore, for `ftp` to work, the NAT gateway has to intercept the IP address and TCP port values and modify them. Additionally, since NAT requires modifications to the packet headers, security mechanisms like IPsec's Authenticated Headers (AH) [15] cannot be used in conjunction with NAT.

Hence, we advocate a cyber-security infrastructure that not only supports advance negotiation and establishment of a secure session between the endpoints; but also enables the discovery of middleboxes and allows the endpoint to negotiate security requirements and request specific functionality from these middleboxes. Further, it features support for negotiating and establishing end-to-end and/or hop-to-hop security associations. Such a cyber-security infrastructure has applicability to general Internet environments, but our focus in this paper is on virtual supercomputing.

Moreover, the concept of a virtual supercomputer provides support for multiple access points. This allows ubiquitous access to virtual supercomputing services, providing users with services anytime and anywhere using low-end machines or even mobile devices, like handheld computers and mobile phones. This puts an additional burden on the cyber-security infrastructure, requiring it to be able to adapt to environments with scarce resources and evolve once more resources become available. Additionally, the infrastructure should not be inherently dependent on IP as there exist many other environments that do not necessarily use IP for communication, e.g., wireless telephony and distributed sensor networking [7].

The remainder of this paper is divided as follows. Section 2 provides some background information on existing security solutions. In Section 3 we mention the limitations of existing security mechanisms. In Section 4 we discuss the requirements for a cyber-security infrastructure for virtual supercomputing. Next, we discuss our system's architecture in Section 5, followed by a brief description of our prototype implementation in Section 6 and conclusions in Section 7.

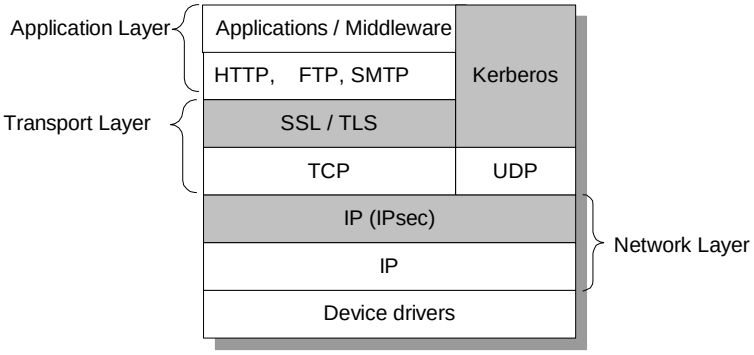

Figure 1: Relative Location of Security Mechanisms in the Protocol Stack

2. Background

Several security mechanisms have been devised mainly to retrofit security into the Internet and existing distributed systems. A security mechanism defines and implements a set of protocols for preventing or detecting security threats. Kerberos [19] is a security mechanism developed at MIT during the mid-1980's to provide users with a single sign-on to the network and protect authentication information from masquerading. Kerberos, however, only supports symmetric cryptography; hence, it does not scale well to large distributed environments. As a result, a number of Kerberos extensions have surfaced. SESAME (Secure European System for Applications in a Multi-vendor Environment) [13] is an extension to Kerberos that provides additional services, such as the use of digital signatures for login and the handling of access control privileges.

While Kerberos and similar technologies provide comprehensive security services at the application layer, there are other security technologies that concentrate on particular components of the network. SSL (Secure Socket Layer) or TLS (Transport Layer Security) [5] implements security just above TCP. SSL makes use of TCP to provide a reliable end-to-end secure service. IP security or IPsec [15] provides improved security to the Internet Protocol. This includes authenticating IP packets and providing data confidentiality and authentication to the IP packet's payload. Figure 1 illustrates the relative location of the above security mechanisms in the protocol stack. Note that the shaded boxes in Figure 1 represent the security mechanisms.

GSI (Grid Security Infrastructure) [10] is a security architecture for grids that can support several underlying security mechanisms.

3. Limitations of Existing Systems

Many of the security approaches mentioned in the previous section offer security as an all-or-nothing option. In some cases, "exportable" security provides a lower level of security for use over international boundaries.

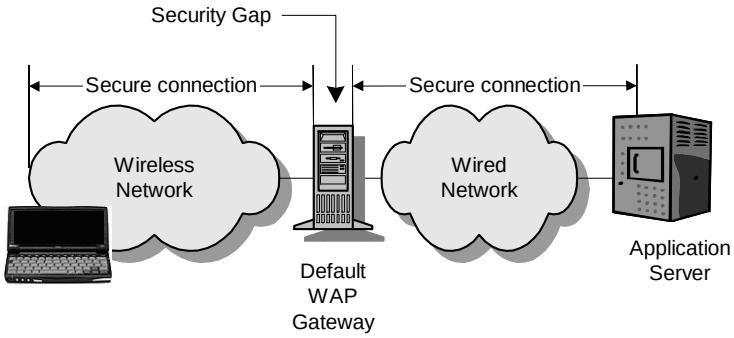

Figure 2: The WAP Security Gap

Another common problem, particularly in the use of SSL and IPsec tunneling over different realms, is the introduction of *security gaps* as described by Ashley et al. [1]. Ashley et al. note that security gaps appear when a secure session is terminated prematurely. A common example of that is found in the Wireless Application Protocol (WAP) [23], which is a standard for delivering information and services to wireless devices such as mobile phones and handheld devices. WAP-enabled devices can access Internet services through a WAP gateway. For secure connections, a secure session is established between the wireless device and the WAP gateway (Figure 2). An SSL session is then established between the WAP gateway and the application server providing the requested service. Due to the termination and reestablishment of the secure sessions, data resides in an insecure state on the WAP gateway.

We argue that security gaps are introduced in any secure path going through one or more middleboxes that need to perform some processing on passing data packets. For example, when IPsec tunneling is employed to secure communications between two networks or gateways (rather than end-to-end) as illustrated in Figure 3, at lease two security gaps are introduced. There are several reasons for doing this. First, it reduces the administrative overhead that results from configuring IPsec on individual machines. Second, it allows the middleboxes that often exist at the boundaries of local networks to function as expected. These include NAT gateways, packet or content filters, proxy firewalls, and WAP gateways.

While this type of IPsec deployment protects data transmission over the Internet, it provides no protection against malicious insiders. Malicious insiders can exploit information destined to inside users and leak it outside the local network. In a survey of 1,225 information security managers, the greatest threat to their networks was virus infections followed by abuse of access privileges by employees [3].

It is also worth noting that IPsec only provides mandatory security. That is, the security policies and attributes are defined by the organization's security officer or system administrator. End applications and users cannot customize the security policies and attributes to meet their particular needs.

While GSI [10] provides secure authentication and communication for grids, it does not attempt to discover middleboxes and negotiate security with them. As a result, security gaps could surface, particularly in cases where some grid resources and nodes exist in a local network behind a firewall. Further, the adaptability of GSI is limited making it hard to port it to lightweight devices with limited capabilities.

4. Requirements

We advocate a cyber-security infrastructure that incorporates greater flexibility, adaptability, and customizability. When it comes to security, one size does not fit all. Hence, the security architecture deployed must be able to adapt to environments with extreme conditions and scarce resources, as well as evolve and provide additional functionality when more resources become available.

Further, with many different security technologies surfacing and being deployed, the assumption that a particular security mechanism will eventually prevail is flawed. For that reason, it is necessary to support multiple security mechanisms and negotiate security requirements. We also aim to reduce or eliminate the security gaps mentioned in the previous section.

While it is necessary to have some security policies mandated by the organization to meet its goals and provide a uniform level of information assurance, end applications

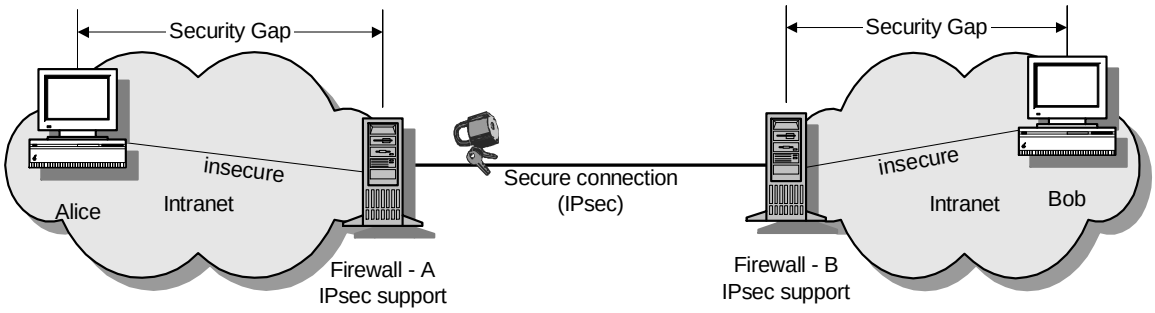

Figure 3: Security Gaps in Current IPsec Implementations

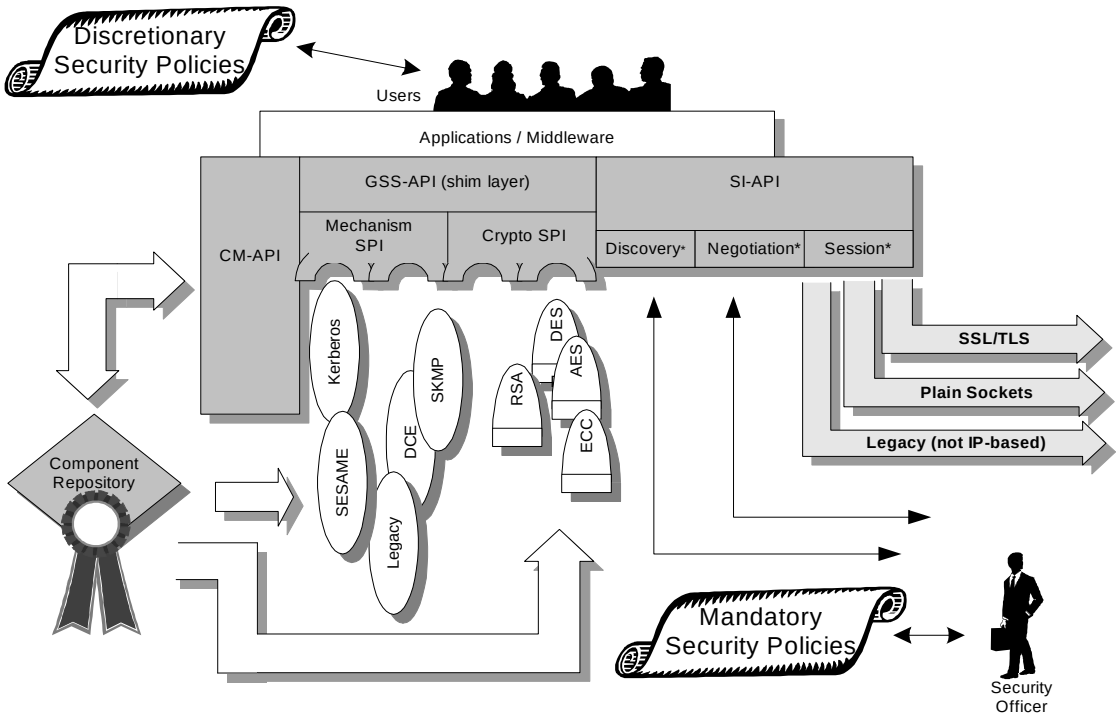

Figure 4: System Design

or users should be able to specify discretionary security and define their own security policies depending on the sensitivity of the information they wish to transmit. Additionally, discretionary security provides an additional line of defense in case the mandatory security is subverted due to an error on behalf of the system administrator, for example.

5. System Architecture

In addition to meeting the requirements mentioned in Section 4, we aim to base the security services in our system on available and proven technologies, thereby granting us flexibility and secure interoperability with existing systems. We, however, plan to enhance these technologies with a component-based design of a cyber-security infrastructure that includes the discovery of middleboxes and the negotiation of security requirements. Figure 4 illustrates the system's architecture, which we describe in this section.

5.1 The Lightweight Core

The system exports three sets of APIs that are available for applications' use. These APIs are GSS-API (Generic Security Services API [17][18]), CM-API (Component Management API) and SI-API (Session Initiation API). These three components together make up the lightweight "core" of the system. This lightweight core would be pre-loaded in all devices. Additional functionality and capabilities could then be loaded on demand into this core, allowing it to expand and evolve.

5.2 GSS-API

The GSS-API exports a uniform, generic interface for providing security services from an underlying security mechanism. The GSS-API functions at a level independent from both the underlying security mechanism and programming environment. This abstraction enables security mechanisms to be removed, added, and updated without affecting the applications. Moreover, unlike SSL and IPsec, GSS-API security services are independent from the communication protocol suite being used. GSS-API has already been used with many security mechanisms including Kerberos, SESAME, SPKM, DCE and others [2]. The independence of GSS-API from underlying mechanisms permits extending the GSS-API to support an arbitrary list of underlying mechanisms. Further, this can be done dynamically such that the necessary security mechanisms or cryptographic functions are loaded on demand. Hence, only the functionality needed at a certain time is loaded. Additionally, this facilitates the incorporation of new security technologies and bug fixes as they become available.

To support several security mechanisms at once and to be able to dynamically query available mechanisms and load/unload them as necessary, we follow the guidance of [21] in which the GSS implementation consists of two parts: (1) the GSS-API shim layer, providing no security itself, but exports the standardized interface, and (2) underlying security mechanisms and supporting cryptographic profiles are added to implement the actual security services.

In between these two layers, there exists SPIs (Service Provider Interfaces) that allow the GSS to locate and query the different security mechanisms and cryptographic functions.

5.3 Loadable Components and the Component Repository

Security mechanisms and cryptographic functions are implemented as add-on components. These components are not part of the pre-loaded lightweight core. Rather, they are loaded whenever their functionality is needed. Alternatively, a machine with sufficient resources that is expected to request many security services and establish several security associations with other devices, a firewall for instance, may choose to preload the necessary functionality as soon as it is bootstrapped. The component-based architecture accommodates both cases and boosts the adaptability of the system, allowing it to unload unnecessary functionality to compensate for shortages in resources and to reload that functionality once resources become more available.

The components that implement additional functionality are stored in a component repository. This repository is certified by a trusted certificate authority to prevent the loading of malicious components. A trusted repository should digitally sign its components to prove their authenticity and protect them against tampering. Component repositories can be stored locally or remotely.

5.4 CM-API

Since the various functionalities (including the security mechanisms and cryptographic libraries) are represented as components, a Component Management API is provided to facilitate the management, loading, unloading and reconfiguration of components as well as the validation of component repositories. The CM-API exposes an interface for the security-aware applications to manipulate and manage available components. The CM-API is also utilized by the GSS-API to load and unload needed functionality on demand.

5.5 SI-API

Session Initiation API (SI-API) provides a session-layer library for applications. It can be used for the discovery of middleboxes and for negotiating security requirements between these middleboxes. Section 5.7 describes the discovery process. This API can be used to establish a communication session. This communication session can be based on plain sockets (in conjunction with GSS-API services to secure transmitted data) or it can use SSL for interoperability with existing systems.

The SI-API exports a simple session library for applications' use, as the following pseudocode illustrates:

```
  //sample application
…
s = session_new();
  // callback function to handle
  // authentication, etc.
session_register_ui_callback(s,
               &my_user_prompt);
session_connect(s, uri, policy, RELIABLE |
               CONGESTION | HOP_ENCRYPTION);
sock = session_fileno(s);
  //if we need to send an address to the
  //other end
t = session_get_my_opaque_addr_token(s);
write(sock, t);
write(sock, …);
…
```

In the pseudocode above, `session_connect()` triggers the necessary discovery and negotiation and calls the UI callback as needed, for authentication and other purposes.

5.6 Security Policies

Implementing state-of-the-art security mechanisms and technologies is not enough to construct a truly secure computing environment. Due to the complexities found in both the physical and virtual worlds, cyber-security infrastructures should take into considerations the various principals that populate a particular domain and the complex relationships among these principals. These principals may include files, objects, resources, machines, and people with different ranks, security attributes and roles. A truly secure environment must define well-structured rules and practices for regulating and managing principals and the complex relationships between them.

In our system, we support two types of policies: mandatory and discretionary. Mandatory policies, which are set forth by the administrative domain's security officer, reflect the organization's security goals. These policies generally dictate what the minimum levels of security are for the administrative domain, which mechanisms are allowed to be used by which users, which component repositories are allowed and which are forbidden, etc. It is possible to assign users different roles and security attributes and lay down different policies based on these attributes.

While it is essential to lay down a system-wide mandatory policy, a discretionary policy allows users and end applications to define additional security requirements. Supporting discretionary policies is important because, in many cases, only users can assess the real value of data being received or transmitted; and therefore, the users are in the best position to determine the necessary level of security beyond mandatory security. Further, it is counterintuitive to apply the same security services on all types of transmitted data. If conflicts between the mandatory and discretionary policies arise, then the more restrictive policy takes precedence.

Users or applications can have policies that specify which loadable components are preferred and which are prohibited, as well as associating particular policies with certain protocols.

5.7 Discovery of Middleboxes

The discovery of middleboxes can be achieved by employing a session signaling protocol as suggested in [8]. Requirements for the discovery of middleboxes are addressed in [16]. We give an overview of how such a protocol works in our infrastructure.

If two endpoints (*A* and *B*) decide to communicate securely, the sender *A* initiates the session-layer signaling protocol by sending a special "discover" request along the path to the end destination (see Figure 5). In this scenario, we assume that endpoint *A* supports security mechanism *X*. Middleboxes along the path reply to the "discover" request to indicate their presence and their security requirements, if any. In the example depicted in Figure 5, the first middlebox along the path is a firewall. If this is the firewall at the boundary of endpoint *A*'s realm it will probably have particular security requirements for outbound traffic. In the example shown, the firewall supports mechanisms *Y* and *Z* and requires all outbound traffic to use one of these mechanisms. The Firewall responds to the "discover" request announcing its existence and the supported security mechanisms (*Y* and *Z* in this case). Since there are no common mechanisms between endpoint *A* and the firewall, only partial negotiations take place at this time. Proceeding with the discovery process, eventually endpoint *B* will respond and communicate its presence and supported mechanisms (*X* and *Y*). Now, endpoint *A* can negotiate the establishment of two layers of security: an end-to-end security session established at endpoint *A*, using mechanism *X*, with the intention to terminate only at endpoint *B*. The other layer is another secure session that is established at midpoint 1 (the firewall) until endpoint *B*, using mechanism *Y*, to meet the additional security requirements imposed by the firewall on outbound traffic. Note that no security gaps are introduced in this scenario.

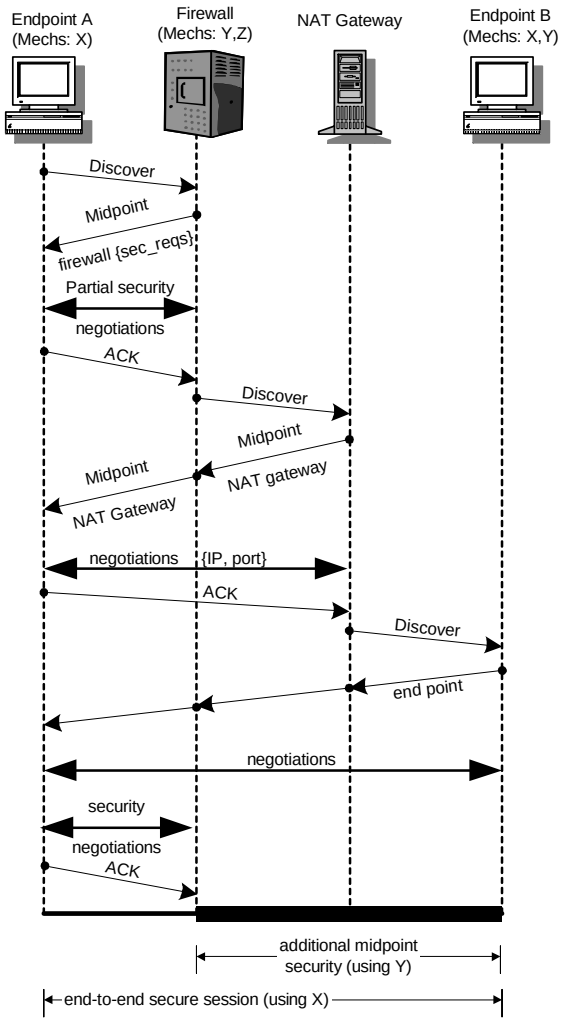

Figure 5: Discovery of Middleboxes

Similarly, some networks may be protected by firewalls or proxies that require certain forms of authentication before network traffic can pass through them. The discovery protocol can locate such firewalls and negotiate a sufficient GSS-API based authentication mechanism.

Figure 5 also depicts a scenario in which a NAT gateway exists somewhere along the path. Likewise, the NAT announces its presence, and negotiations kick-in during which the destination IP address and port are communicated, allowing the NAT gateway to deliver the data to the original destination without introducing security gaps (through tunneling, for example). Alternatively, RSIP (Realm Specific IP) [4] and similar protocols can be employed in place of NAT. RSIP is a proposed standard that allows a system to allocate a global address from a local gateway. The system then uses that global IP for outside communications but tunnels the traffic to the local gateway. In the reverse direction, the gateway maintains a table of allocated addresses and tunnels incoming traffic to the appropriate local system.

6. System Implementation

We have implemented a prototype of the described cybersecurity infrastructure. We refer to the prototype implementation as IRIS (Inter-Realm Infrastructure for Security). Our implementation is in Java. We have used Java for its cross-platform compatibility, which allows IRIS to be ported easily to the heterogeneous nodes that constitute a virtual supercomputer. Additionally, Java is gaining grounds in the mobile world allowing IRIS to be ported to mobiles, handheld computers and embedded devices.

In our current prototype, the component repository is implemented as a standalone server on the network. Components can be queried and downloaded remotely. Since

component repositories can be distributed on a global scale, we employ Jini technology [14], which provides an infrastructure for discovering and delivering services in distributed computing environments. The Jini-based repository in IRIS allows service providers to write security mechanisms and cryptographic functionality as certified components that can be loaded remotely by applications.

Our implementation of the GSS-API is based on the RFC that defines the Java binding for the standardized GSS-API [12], whereas the SPI is based on [21]. We have defined our own proprietary interfaces for the CM-API and SI-API.

7. Conclusion

The full potential of virtual supercomputing can only be realized once an adequate, inter-realm, cyber-security infrastructure is devised and deployed. We discussed the requirements of such an infrastructure and designed an architecture that provides comprehensive security services in a component-based fashion, providing greater adaptability, customizability and backward-compatibility. We briefly described our prototype implementation of the system. Finally, we note that although this infrastructure was described in the context of virtual supercomputing, the infrastructure can be employed to secure other networked environments.

8. References

[1] P. Ashley et al., "Wired versus Wireless Security: The Internet, WAP, and iMode for E-Commerce," submitted to the *17th Annual Computer Security Applications Conference (ACSAC 2001).*

[2] P. Ashley and M. Vandenwauver, "Practical Intranet Security Overview of the State of the Art and Available Technologies," Kluwer Academic Publishers, 1999.

[3] D. Bernstein, "Infosecurity News – Industry Survey," *Infosecurity News*, 8(3): 22-27, May 1997.

[4] M. Borella and J. Lo, "Realm Specific IP: Framework," Internet Draft <draft-ietf-nat-rsip-framework-05.txt>, July 2000. Work in progress.

[5] T. Dierks and C. Allen, "The TLS Protocol," January 1999, RFC 2246.

[6] K. Egevang and P. Francis, "The IP Network Address Translator (NAT)," May 1994, RFC 1631.

[7] J. Elson and D. Estrin, "Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks," *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, Phoenix, Arizona, April 2001.

[8] M. Fisk and W. Feng, "Interactions of Realm Boundaries and End-to-End Network Applications," Los Alamos Unclassified Report (LAUR) 00-3631.

[9] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann Publishers, January 1998.

[10] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, "A Security Architecture for Computational Grids," *Proceedings of the 5th ACM Conference on Computer and Communication Security*, 83-92, 1998.

[11] Intel Corp., "Peer-to-Peer: Spreading the Computer Power," online article, http://www.intel.com/eBusiness/products/peertopeer/ar011102.htm

[12] J. Kabat et al., "Generic Security Service API Version 2: Java Bindings," June 2000, RFC 2853.

[13] P. Kaijser, T. Parker, and D. Pinkas, "SESAME: The Solution to Security for Open Distributed Systems," *Computer Communications*, 17(7): 501-518, July 1994.

[14] W. Keith Edwards, "Core JINI," Prentice Hall PTR, 1999.

[15] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol," November 1998, RCF 2401.

[16] E. Lear, "Requirements for Discovering Middleboxes," Internet Draft <draft-lear-middlebox-discovery-requirements-00.txt>, April 2001. Work in progress.

[17] J. Linn, "Generic Security Service Application Program Interface," September 1993, RFC 1508.

[18] J. Linn, "Generic Security Service Application Program Interface Version 2," January 1997, RFC 2078.

[19] B. Neumann and T. Ts'o, "Kerberos: An Authentication Service for Computer Networks," *IEEE Communications Magazine*, 32(9): 33-38, September 1994.

[20] J. Postel and J. K. Reynolds, "File Transfer Protocol," October 1985, RFC 959.

[21] M. Smith, "A Service Provider API for GSS mechanisms in Java," Internet Draft <draft-ietf-cat-gssv2-javabind-spi-02.txt>, October 1999. Work in progress.

[22] P. Srisuresh and M. Holdrege, "IP Network Address Translator (NAT) terminology and considerations," August 1999, RFC 2663.

[23] The WAP Forum, http://www.wapforum.org