Design of a Simple, Distributed Network Access Control System

By David Boen, Daniel Dent, Victor Chan, Andrew Tjia

Abstract—Network access control describes the measures used to control network nodes prior to joining and after admission into the network system, using a combination of security policies and controls. This paper describes the design and implementation of the Distributed Firewall approach to network access control, with a focus on security policy enforcement. We compare our method with alternative forms of network access control. Finally, we test our system using real-life applications and demonstrate its effectiveness.

Index Terms—Distributed Network Access Control, Distributed Firewalls, Security Policy Enforcement

I. Introduction

Networks require secure access control measures to prevent malicious users from consuming resources. Various schemes have been devised by developers and implemented for many Internet applications from telephony, to secure online commerce, or to data storage systems. This is typically solved through a combination of network access control measures. Network access control (NAC) describes a set of secure protocols used to secure nodes prior to joining and control methods to limit access after joining other nodes on a network. Network administrators define policies that new or existing nodes must obey. However, network access controls are difficult to enforce over a geographically separated enterprise network. A simple, uniform solution is needed that scales over multiple machines and reduces the points of failure.

One of our group members is currently employed as a system administrator for a number of corporate clients. His findings indicate that remote administration is problematic due to constantly changing network access policies. As a result, much work is needed to update and manage the policies on multiple sites. In his opinion, a centrally managed solution would simplify his network access control issues.

A distributed firewall allows a security policy to be centrally managed, but distributes enforcement to the individual end nodes. A distributed firewall consists of number of components as defined in [1]:

1) A protocol to define permitted/unpermitted connections in a policy.
2) A secure means of distributing the policy (such as IPSec).
3) A mechanism to interpret and enforce the policy on the end machine.

We feel that a distributed firewall offers many advantages over the conventional perimeter firewall and VPN solutions to network access control [2] [3]. In this paper, we focus on the interpretation and policy enforcement mechanism of a distributed firewall.

This paper is organized as follows: Section II elaborates on various network access control methods currently available, while Section III highlights the advantages of the DNAC method. Sections IV and V describe our design and implementation of a simple DNAC system,

II. Related Work – Conventional Network Access Methods

A firewall is a collection of components interposed between two networks that filter traffic between them according to some defined security policy [3]. Users and hosts connected behind a firewall are assumed trustworthy. Today, a large number of individual firewalls may be used to enforce a consistent policy. The distribution and maintenance of this policy over separate firewalls is responsible for much complexity.

Figure 1 illustrates a conventional firewall configuration, involving a remote, mobile user connecting through the Internet to an internal network.

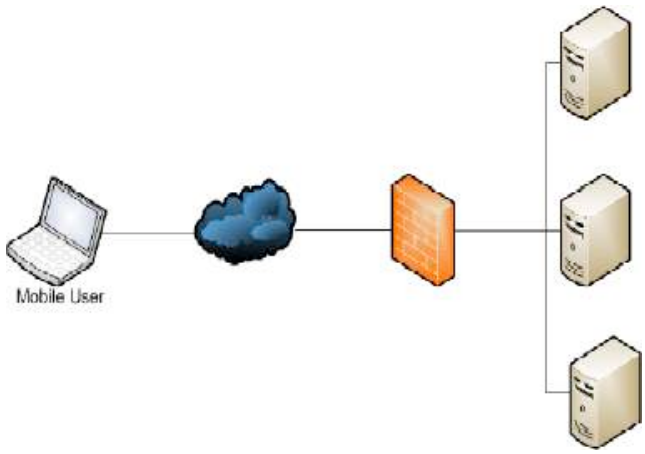

Figure 1 – A conventional firewall configuration

A. Conventional Perimeter Firewalls

First, we will consider a simple network controlled by a single firewall. Firewalls rely on restrictions on network topology to filter traffic between two networks. The single

firewall controls access to the network through a centralized point. Firewalls range from simple packet filtering devices to more advanced state-aware and application aware devices [4]. Simple packet filters make decisions based upon the contents of a single packet, while stateful filters utilize the data of the entire stream or connection. Application aware devices are able to make decisions at a higher level. Nevertheless, with all firewall devices, complex rules must be defined to suit the many different users and requirements behind the firewall. A key assumption is that all users behind the firewall are trusted.

Enforcement of the firewall rules occurs at the firewall. Thus, a conventional firewall represents a point of failure on the network. Due to the increasing speed and volume of data traffic, firewalls represent performance bottlenecks. In addition, if data is analyzed at the packet level to increase performance, conventional firewalls lack the flexibility to suit application specific needs, such as quality-of-service or performance characteristics.

B. Personal Firewalls

Personal firewall software, such as Windows Firewall, ZoneAlarm or Linux *iptables* offer a simple and cheap solution. However, both policy definition and enforcement are left to the end user's discretion.

C. Virtual Private Networks

Virtual Private Networks (VPNs) have emerged as a cost effect and widespread means to provide secure communication across geographically distributed networks. A VPN securely tunnels data between two private end-points over a public network such as the Internet. Today, implementations, such as PPTP and IPSec, are included in many major operating systems. However, the end-to-end encryption prevents network firewalls from reading the network traffic, defeating their data filtering capabilities. Also, an end user connected via VPN is trusted as if they were internally connected to the internal private network, with unrestricted access. Access policies are statically set until the connection is terminated.

III. Distributed Network Access Control

A distributed network access control (DNAC) method offers a number of distinct advantages over conventional access control methods, while retaining correcting some of the shortcomings. Table 1 compares conventional firewalls and the distributed network access control method.

Table 1: A Comparison of conventional firewalls with Distributed Firewalls

Factor	Conventional Firewall	Distributed Network Access Control
Network Topology Dependence	Only users included in the network topology are protected.	Remote users are protected.
Policy definition	Decentralized	Centralized
Policy distribution.	Static.	Dynamic. Administrators can push policy updates to machines automatically
Policy granularity	Coarse-grained (general user needs)	Fine-grained (application specific)
Trust	All users are trusted.	Users may be individually assigned trust levels.
Performance and availability	Dependent on firewall hosts.	Workload distributed to destination machines.
Scalability	Requires additional hardware	Does not require additional hardware. Filtering performed on destination machines.

Distributed Network Access offers distinct advantages over conventional firewalls and VPNs in the area of policy management and distribution, customized user trust and performance scalability.

Figure 2 illustrates a configuration of a distributed firewall.

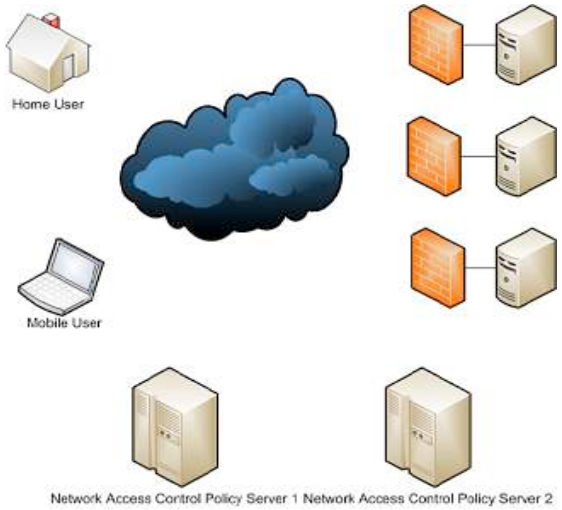

Figure 2: An example of Distributed Network Access Control layout

IV. Design

A. Design Objectives

We designed our system with the following features in mind:

- Scalability – The system allows additional servers to be added to improve performance.
- Availability – Multiple servers provide access to the Internet and authorization mechanisms.

Consequently, we devised three main entities to manage the user access and authorization. Table 1 highlights their associated responsibilities, as listed in Table 2.

Table 2: The Distributed Firewall Entities and associated responsibilities

Entity	Responsibility
Policy Server	• Maintains a log of actions taken for auditing purposes. • Authenticates Clients • Authorizes Clients to access resources on Protected Hosts. • Forwards authorized Client Requests to the Protected Hosts.
Protected Host	• Maintains a log of actions taken • Applies Firewall Transitions (FWT) as requested by the Policy Server • Deals with expiring leases • Services authorized Clients with access to resources (ports).
Client	• Maintains a log of actions taken. • Maintains a list of firewall rules • Maintains identity info for Authentication • Makes Policy Server requests.

Firewall Transitions (FWT) describe a set of ports opened or closed to support the application requested by a Client. A Client requests a FWT from the Protected Host through the Policy Server. The Policy Server grants the request if the Client is authorized. The Protected Host will then work with the application data sent by the Client and relayed by the Policy Server. Figure 3 illustrates the interaction between the entities in our architecture.

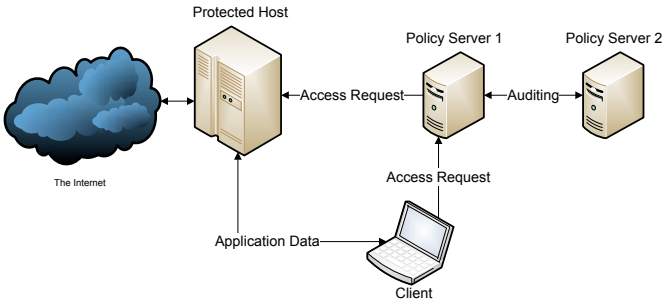

Figure 3: The Distributed Firewall Architecture

V. Implementation

A. Functional Blocks

1) Client

The Client was implemented using Java using a Model-View-Controller approach with the core protocol stack shared between the client and server. It allows client side configuration such that the user can create and store saved connection setups complete with server and authentication details, which are serialized as an XML configuration file. This XML configuration file can be distributed by a system administrator for easy deployment across an enterprise.

After checking that the user has a syntactically correct request, the client packages it into a Client Access Request message and fires it to the Policy Server.

2) Policy Server

The Policy Server is implemented using Java with a three-tier approach to design. It provides services to the clients and accepts client requests by authenticating the origin and consulting a backend database for user, host, and permissions lookup.

The backend database is accessed via JDBC and is a Microsoft Excel spreadsheet for easy editing of policy. Other users may opt to rewrite the database layer through implementing the provided interfaces and use an SQL DBMS backend instead.

Auditing is implemented via a logging system, *Log4J*, which allows auditing in varied ways. The simplest form is output to a flat file and is the strategy used for this implementation. Auditors and other interested parties can use search tools such as *grep* in order to find interesting records.

When the policy server receives a Client Access Request message from a client, it spawns a separate thread to deal with the request. This thread is tasked with decrypting the message with the server's private key, verifying the message through its embedded signature matched with the client's public key, and ensuring that the client has permission to make the firewall transition request. When all parameters have checked out successfully, the server sends a single UDP packet over secure channels to the protected host to negotiate the firewall transition.

3) Protected Host

The Protected Host is implemented as a daemon written in Perl and accepts authenticated Client Request messages from the Policy Server. Using policies distributed through means such as *rsync*, the daemon maps the firewall transition rule with these policies stored as flat files, and generates commands to manipulate the firewall. *Iptables* is integrated within the Linux kernel as of version 2.4 and is manipulated with the user-space command, *iptables*. This command can dynamically modify the kernel firewall tables on the fly. Each request is timestamped and inserted into a database to eventually queue the expiration of the firewall rule at the end of the lease period. A *cron* job runs in the background, consults the queue, and removes the rules corresponding to

expired leases.

The DNAC system is implemented with *iptables* in mind, but in the future, can be expanded to other firewall systems and other operating systems such as Microsoft Windows.

B. Interactions

Table 3 and Figure 4 describe the message exchanges between entities and resulting actions in our system.

Table 3: Messages and their resulting actions

Exchange	Message	Actions
1 a)	Client Access Request	• The Client requests access to an application residing on the Policy Host
1 b)	Policy Server Request for Authorization	• The Policy Server checks the Client's access. • If the Client is authorized to access the application, the Policy Server signs the Client Access Request and forwards request to Policy Host
2)	Client Application Communication	• The Client attempts to access the application resource residing on the Policy Host • If the Client Request is honored by the Policy Server (the client has sufficient rights), the request will proceed.
3)	Mutual Auditing	• Policy Servers exchange log information periodically

C. Communications and Message Format

The message exchanges are transmitted between entities through the use of TCP messages, in the following format.

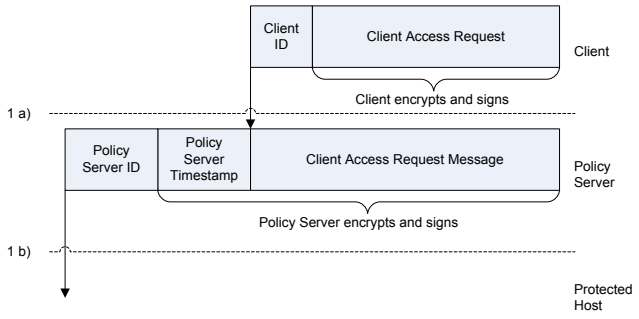

Figure 4: Client, Policy Server, Protected Host message format

Table 4 describes the fields in the Client Access Request message. All the fields are 32-bit integers.

Table 4: The Client Access Request message fields

Field	Description
Client ID	ID of the Client sending the Access Request
Time of Request	Time the client sent the request message
Protect Host ID	The ID of the Protected Host the Client is attempting to access.
FWT Rule	Firewall Transition rule. The defined rule the Protected Host execute (typically port activation/deactivation)
FWTParam	The IP Address of the requesting client.
Lease Time (min)	The maximum time the Client requests the FWT to be active.

The time of request field prevents replay attacks on the system, but requires that the Client and Policy Server clocks are synchronized via Network Time Protocol (NTP). The lease time ensures that Clients are not denied service on the Protected Hosts in a given time window.

D. Confidentiality and Integrity – Encryption

RSA encryption is used throughout the exchanges between the client, policy server and protected host. Public key encryption mechanism is applied to requests sent from the client to the Policy Server and from the Policy Server to the Protected Hosts. Client requests are signed using the client's private key, which is in turn encrypted with the Policy Server's public key. Once the Policy Server processes the client's request, it signs the unencrypted client request and encrypts it with the target's Protected Host's public key. Then, the request is then passed on to the Protected Host. In this design, each component must possess the proper set of keys in order to facilitate the passing of encrypted requests. As such, each Client requires registration at the Policy Server prior to engaging in firewall transition requests.

E. Policy Definition

We simplified our design and eliminated the flexible Policy Definition Language in [1] in favor of a direct *iptables* script.

We defined some standard policies to suit different applications for SSH (port 22), an HTTPS Secure Web Server (port 443), and POP3 Mail Server (port 110). A system administrator can easily define new policies using an *iptables* script.

We do not provide a mechanism for the Client to verify that it has been authorized to access the Protected Host. The Client assumes a connection and tries to access the offered port, but it is up to the Protected Host to allow a connection transmit the Application Data through the Internet.

VI. Related Work And Extensions

The *GreenBow* distributed firewall [5] is an example of a commercial product based on the distributed firewall concept.

In further extensions, it is very likely that new policies will need to be pushed to the Protected Hosts. This requires the use of a distribution mechanism, which may be implemented in many ways, including *rsync* or a customized file transfer protocol. Regardless, a file on the Protected Host requires updating to include the new rules.

Alternatively, it is possible to have the Protected Hosts pull the set of FWTs (stored inside a configuration file) from a given server. This requires rigid security during the exchange. Furthermore, we could develop a formal Policy Definition Language [1] rather than hardcode FWTs in *iptables* scripts, allowing policies to be interpreted on non-Linux platforms.

VII. Conclusion

. We demonstrated a simple, easily managed, and secure system to allow remote clients to request access and gain access to resources on an internal network. We tested the operation of our system using the SSH application (SSH) and demonstrated connections and disconnections. It is expected that enterprise subscribers of a DNACs system will use this system to supplement a growing arsenal of tools against malicious users across a variety of networks.

References

[1] S. Ioannidis. et al., “Implementing a Distributed Firewall”, *In Proceedings of Computer and Communications Security (CCS) 2000*, November 2000. http://www1.cs.columbia.edu/~angelos/Papers/df.pdf

[2] Steven M. Bellovin, "Distributed Firewalls", ;*login:*, November 1999, pp. 39-47. Available at: http://www.cs.columbia.edu/~smb/papers/distfw.html

[3] Christos Douligeris and Dimitrios N. Sepanos. Network Security: Current Status and Future Directions. IEEE Press @ 2007.

[4] Mark Stamp, *Information Security: Principles and Practices.* ., Hoboken, New Jersey. John Wiley & Sons Inc. 2006 pp. 191.

[5] TheGreenBow, Sistech SA. TheGreenBow Distributed Firewall. Available at: http://www.thegreenbow.com/fwd.html