Distributed Firewall
Overview

- Introduction (conventional firewall)
- Distributed firewall
- Components
- Keynotes
- Implementation
  - Kernel
  - Device driver
  - Policy daemon

Example

Conclusion (advantages & disadvantages)
Introduction

- Hardware or software that limits access to outsider (internet).
- Examines the headers of each intercepted packet.
- Decides whether to accept the packet or to discard the packet.
- Rely on network topology restrictions.
- Assumes that everyone inside protected n/w is trusted.
Disadvantages of conventional firewall

- Increased gap between processing and networking speed.
- Difficulty in processing new protocols.
- Assumption that insiders to be trusted.
- Establishing of new entry points without administrators knowledge.
- Difficulty in administration and management.
- Allowing end-to-end encryption can be threat to firewall
- Lack of finer-grained access control.
Distributed Firewall

- In distributed firewall policy is defined centrally.
- System propagates the central policy to all endpoints.

- Policy may be pushed to end systems or provided in the form of credentials, or both.

- Policy defines what connectivity is permitted.

- End points are characterized by their IPSec Identity.
Components Of Distributed firewall

- A language for expressing policies and resolving requests
- A mechanism for safely distributing security policies.
- A mechanism that applies the security policy to incoming packets or connections.
Components Of Distributed firewall

- KeyNote trust-management system is used for Language specification.

- overall security policy is the composition of the security policy “pushed” to the end host, any credentials given to the user, and any credentials stored in a central location and retrieved on-demand.

- IPsec is used for traffic protection and user/host authentication.

- policy distribution:
  - distribute the KeyNote credentials
  - credentials can be pushed directly to the end hosts
  - credentials can be placed in a repository
KeyNote

- provides a simple notation for specifying both local security policies and credentials.
- Policies and credentials contain predicates that describe the trusted actions
KeyNote

- Monotonicity
- uses cryptographic keys as principal identifiers
- Other than cryptographic keys usernames may be used to identify principals inside a host (Controlled by Operating System).
- End hosts are also considered principals when IPsec is not used to secure communications.
- support application-specific credentials.
- Credentials may be considered as an extension, or refinement, of local policy
- merge policies from different administrative entities
Benefits of KeyNote

- It can handle a variety of different applications
- Provides built-in delegation
- Allows for incremental or localized policy updates
Sample KeyNote assertions, a policy and a (signed) credential

- **KeyNote-Version: 2**
  Authorizer: "POLICY"
  Licensees: "rsa-hex:1023abcd"
  Comment: Allow Licensee to connect to local port 23 (telnet) from internal addresses only, or to port 22 (ssh) from anywhere. Since this is a policy, no signature field is required.
  Conditions: (local_port == "23" && protocol == "tcp" && remote_address > "158.130.006.000" && remote_address < "158.130.007.255") -> "true"; local_port == "22" && protocol == "tcp" -> "true";

- **KeyNote-Version: 2**
  Authorizer: "rsa-hex:1023abcd"
  Licensees: "dsa-hex:98 6512a1" || "x509-base64:19abcd02=="
  Comment: Authorizer delegates SSH connection access to either of the Licensees, if coming from a specific address.
  Conditions: (remote_address == "139.091.001.001" && local_port == "22") -> "true";
  Signature: "rsa-md5-hex:f00f5673"
Implementation Of DF

- Implementation OpenBSD operating system
- System comprises 3 components:
  - kernel extensions
  - daemon process
  - device driver
Kernel Extensions

- Users create outgoing and allow incoming TCP connections using the connect() and accept() system calls respectively.

- Filters can be implemented either in user space or inside the kernel.

- User level approach has the advantage of operating system independence.

- Kernel level approach, requires modifications to the operating system kernel.

- Kernel creates a policy context.

- All the information regarding policy is added to the context.

- Commit operation adds the context to the list of contexts the policy daemon needs to handle.
Policy Device (Driver)

- serves as a communication path between the user-space policy daemon, and the modified system calls in the kernel.
  - Supports operations like open(), close(), read(), write(), and ioctl().
  - Implemented as a loadable module

- Opening the device activates the distributed firewall and initializes data structures

- Closing the device will free any allocated resources and disable the distributed firewall.
Policy Daemon

- It is a user level process responsible for making decisions.
- Daemon receives each request from the kernel by reading the device.
- Processing is done using KeyNote Library and a decision to accept or deny.
- Daemon has no awareness of the specific application.
- Fetch credential based on ID of user or local/remote IP address.
Example

- Local host has local policy
- Remote host will have established an IPsec Security Association with the local host using IKE
- KeyNote credential will be-

  KeyNote-Version: 2
  Authorizer: ADMINISTRATIVE_KEY
  Licensees: USER_KEY
  Conditions:(app_domain == "IPsec policy" && encryption_algorithm == "3DES" && local_address == "158.130.006.141") -> "true";(app_domain == "Distributed Firewall" && @local_port == 23 && encrypted == "yes" && authenticated == "yes") -> "true";
  Signature: ...
Example

- kernel will construct the appropriate context for the TCP connection
- Information will be passed to the policy daemon.

- policy daemon will perform a KeyNote evaluation

- positive response – kernel will permit TCP connection to proceed.

- Negative response - policy daemon will try to acquire relevant credentials by contacting a remote server where these are stored
Conclusion (Advantages)

- Security is no longer dependent on restricting the network topology.
- Eliminate a performance bottleneck.
- Filtering of certain protocols (e.g., FTP) becomes easier.
- Outside connections do not cause administration nightmares.
- Insiders may no longer be treated as unconditionally trusted.
- End-to-end encryption is made possible.
- Application-specific policies may be made available.
- Filtering (and other policy) rules are distributed and established on an as-needed basis.
References

- T. Aura.” Distributed access rights management with delegation certificates”.

- Sotiris Ioannidis, Angelos D. Keromytis, Steve M. Bellovin, Jonathan M. Smith “Implementing a Distributed Firewall”.

- P. Senthilkumar, Dr. S. Arumugam, “Allowing and Storing of Authorized and Unauthorized Database User According to the Policy Verification and Validation of Distributed Firewall under the Specialized Database”.

- Steven M. Bellovin, “Distributed Firewalls”.
Thank You