Implementing a Distributed Firewall

Sotiris Ioannidis
Angelos D. Keromytis
Steve M. Bellovin
Jonathan M. Smith

Presented By

Jim Michaud

Outline

- Intro to Security and Firewalls
- Problems with Current Firewalls
- Distributed Firewall Concept
- Distributed Firewall Implementation
- Conclusions

Intro to Security

- Computer/Network Security - The prevention and detection of unauthorized actions by users of computer systems*
- But what does “unauthorized” mean?
- It depends on the system’s “*security policy*”

*Dieter Gollman, Computer Security, p 9

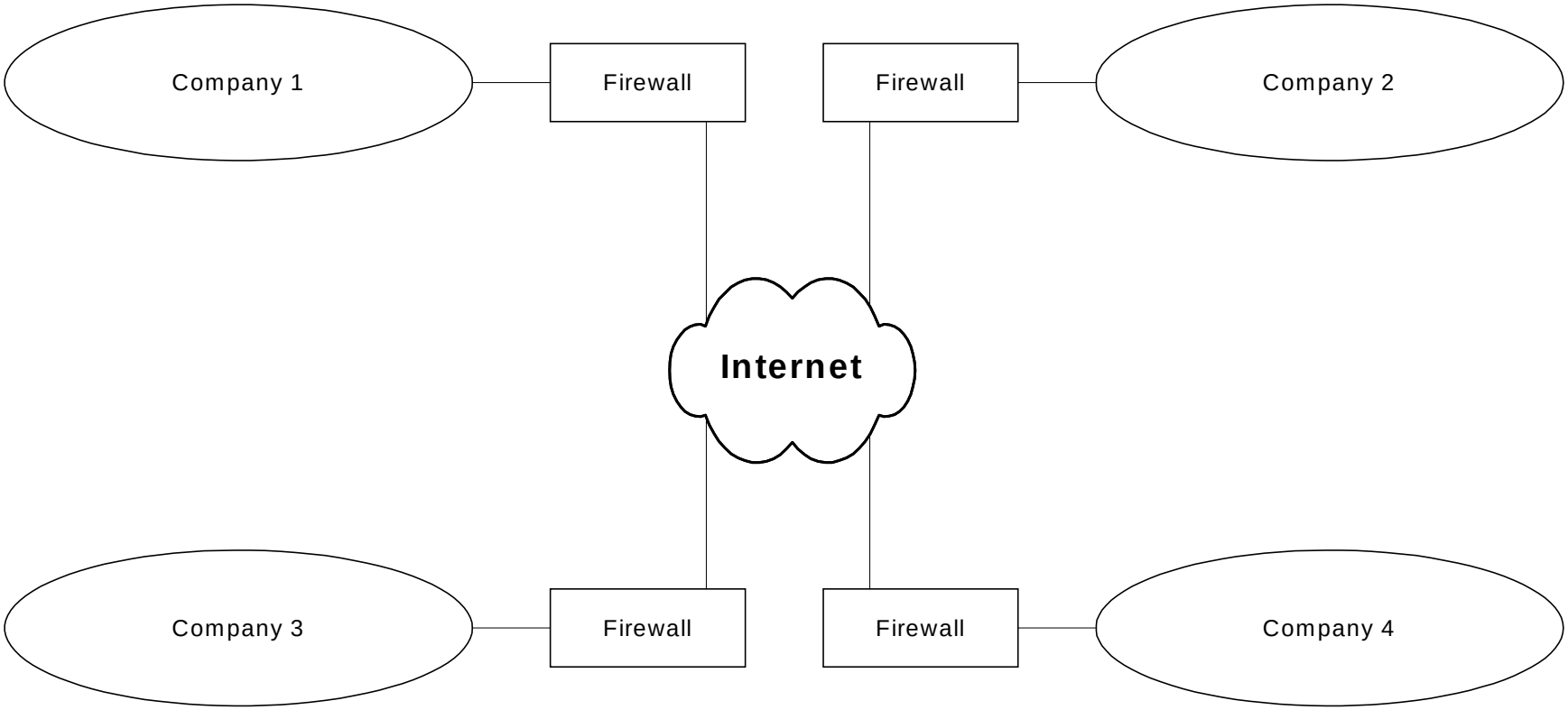
Security Policy

- A "*security policy*" defines the security rules of a system.
- Without a defined security policy, there is no way to know what access is allowed or disallowed
- An example policy: (simple)
  - *Allow all connections to the web server*
  - *Deny all other access*

Firewalls

- In most systems today, the firewall is the machine that implements the "*security policy*" for a system
- A firewall is typically placed at the edge of a system and acts as a filter for unauthorized traffic
- Filters tend to be simple: source and destination addresses, source and destination ports, or protocol (tcp, udp, icmp)

Firewall Example



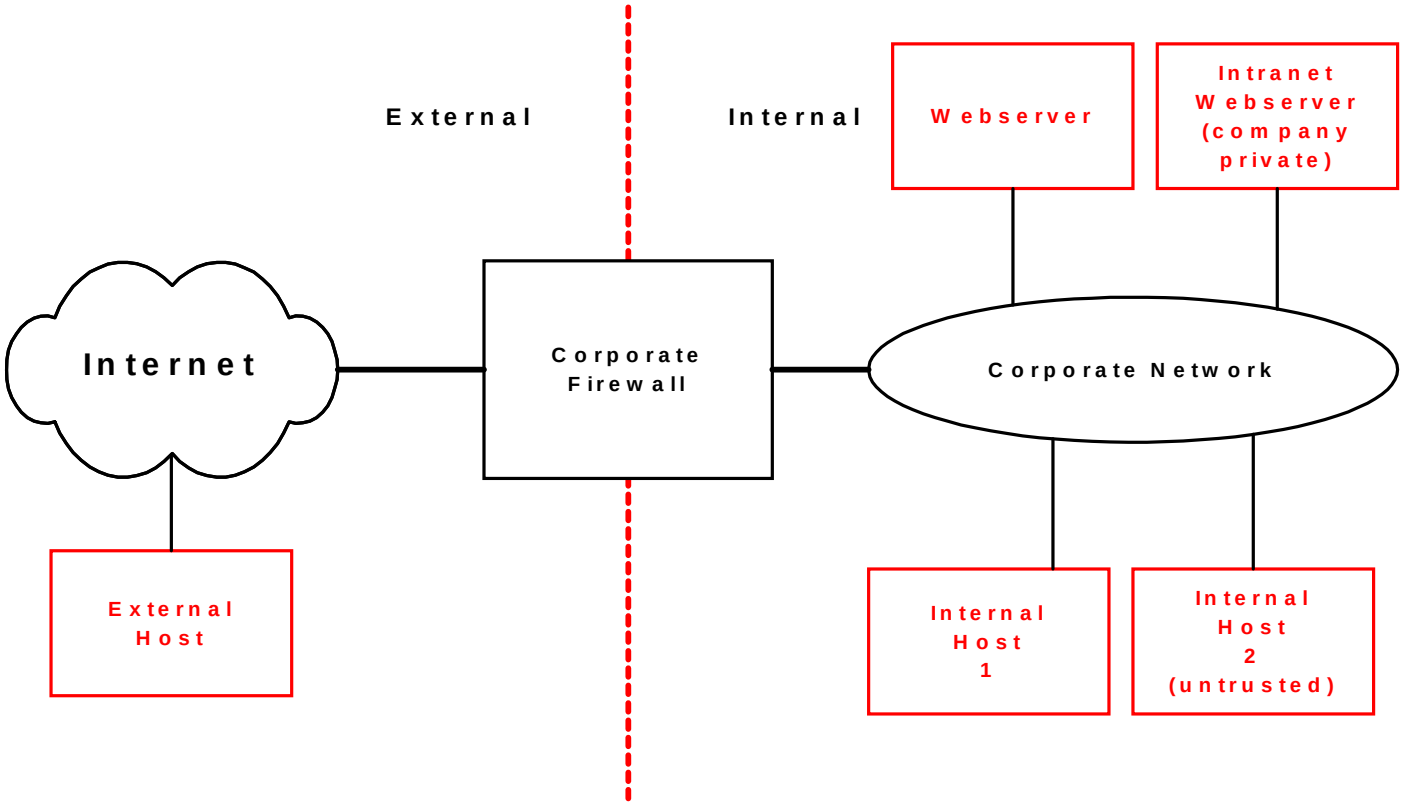
Firewall Drawbacks

- Firewalls can become a bottleneck
- Certain protocols (FTP, Real-Audio) are difficult for firewalls to process
- Assumes inside users are “trusted”
- Multiple entry points make firewalls hard to manage

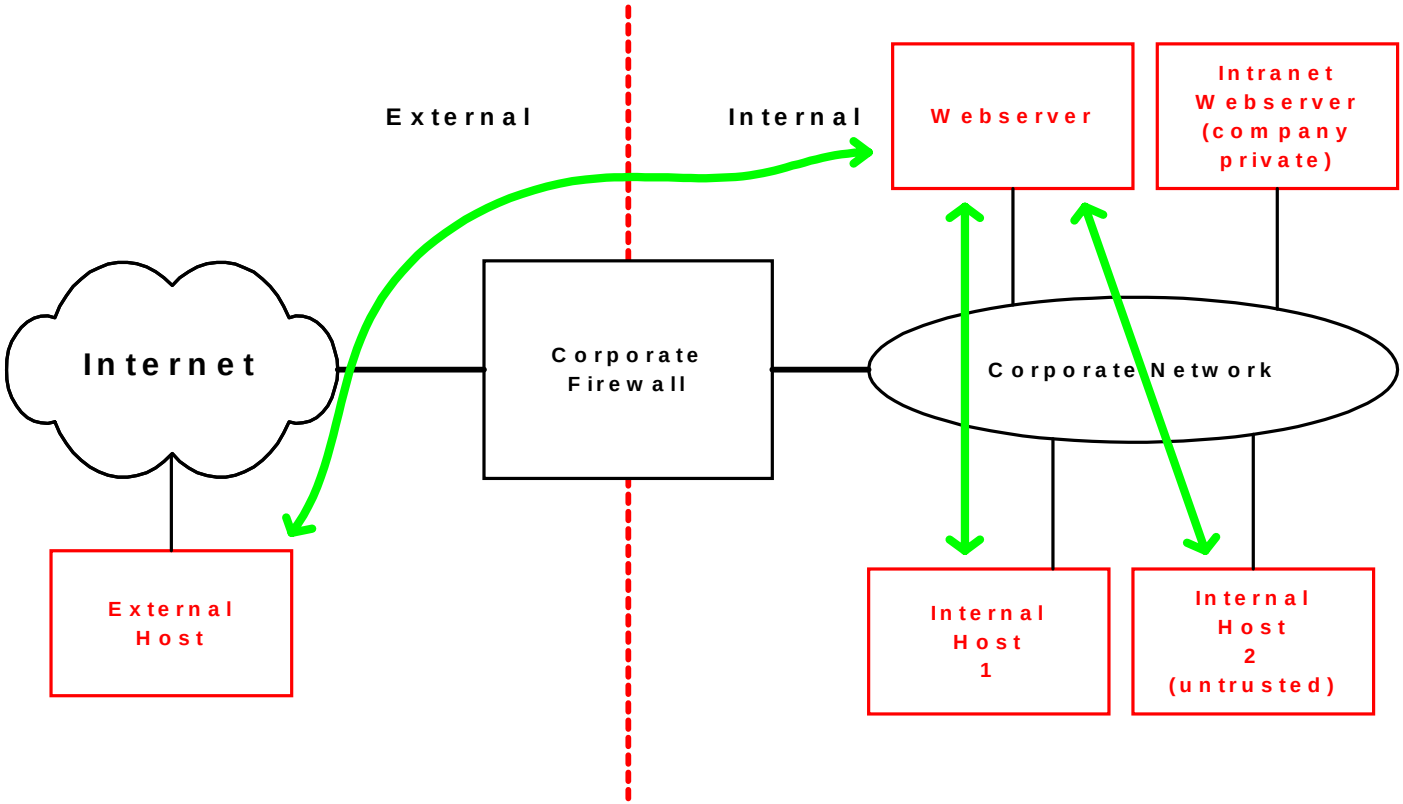
Distributed Firewall Concept

- Security policy is defined centrally
- Enforcement of policy is done by network endpoint(s)

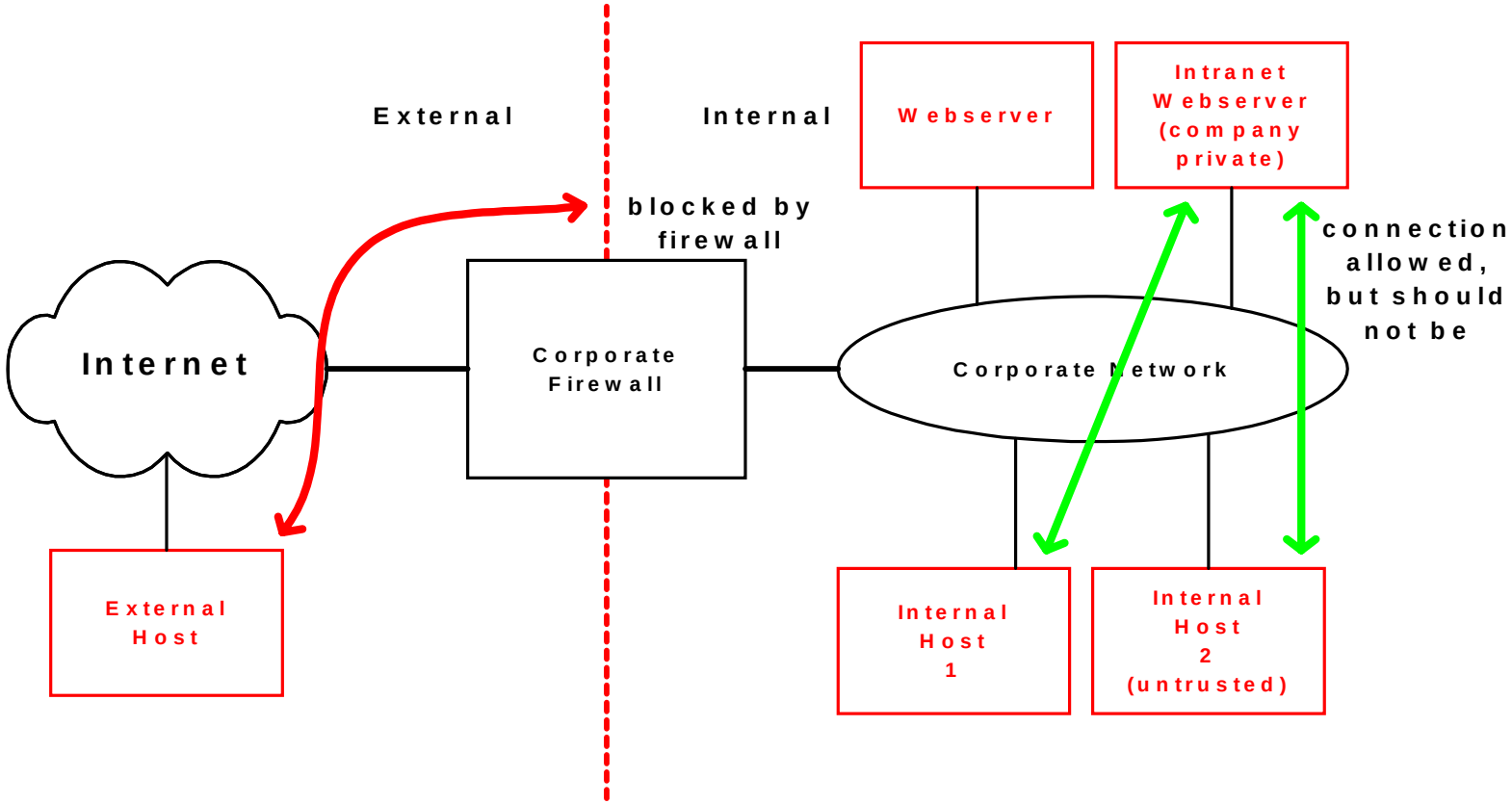
Standard Firewall Example



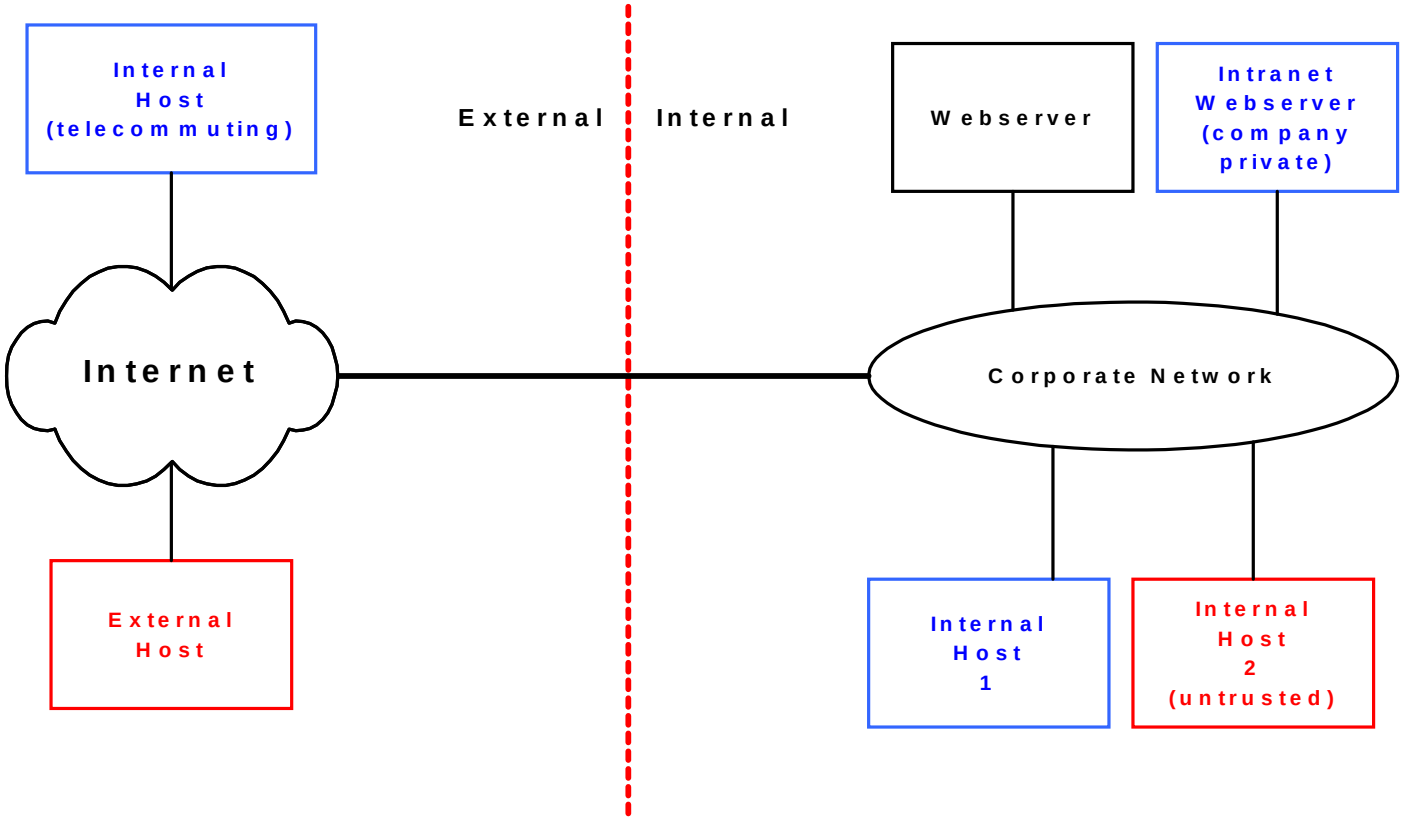
Standard Firewall Example
Connection to web server



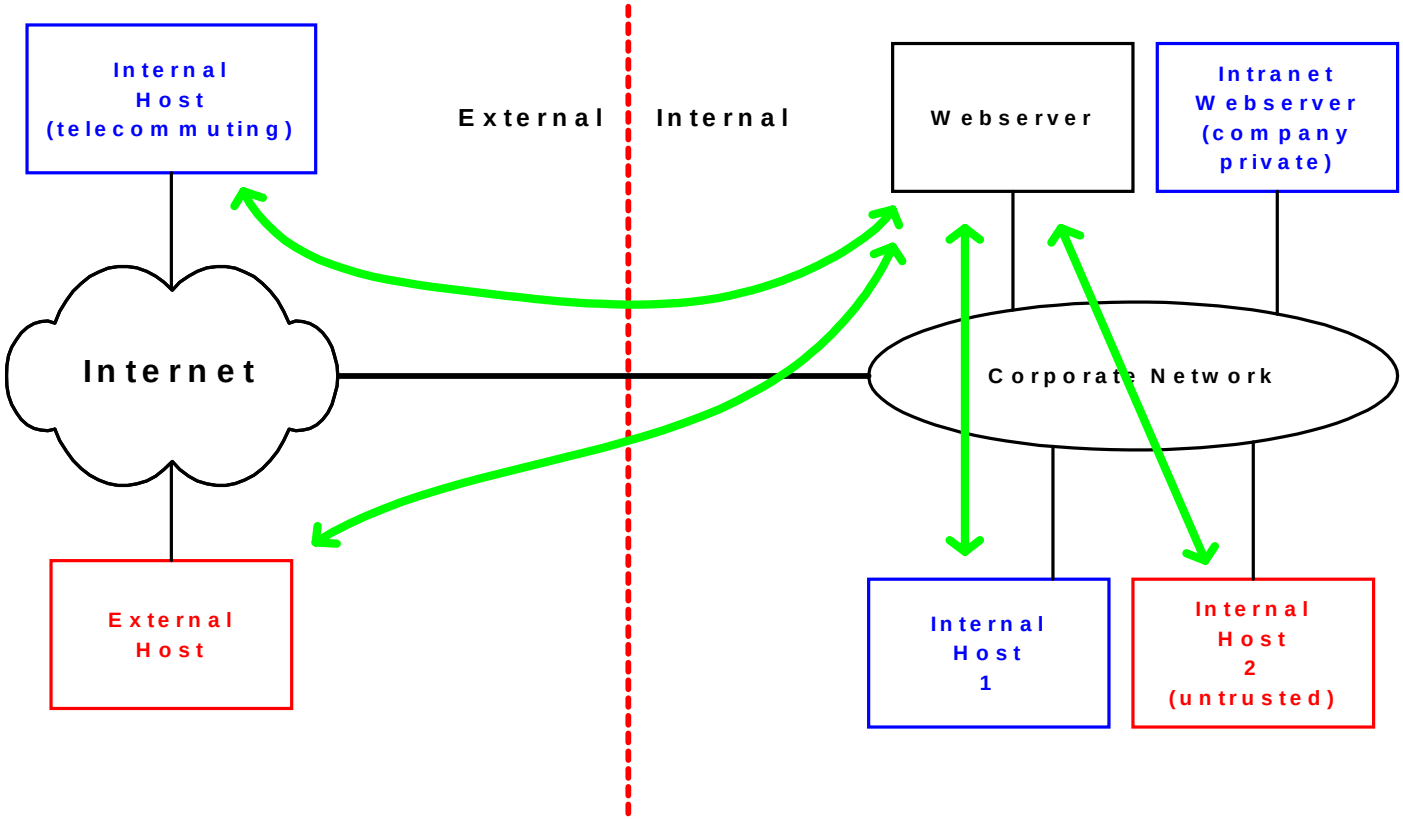
Standard Firewall Example
Connection to intranet



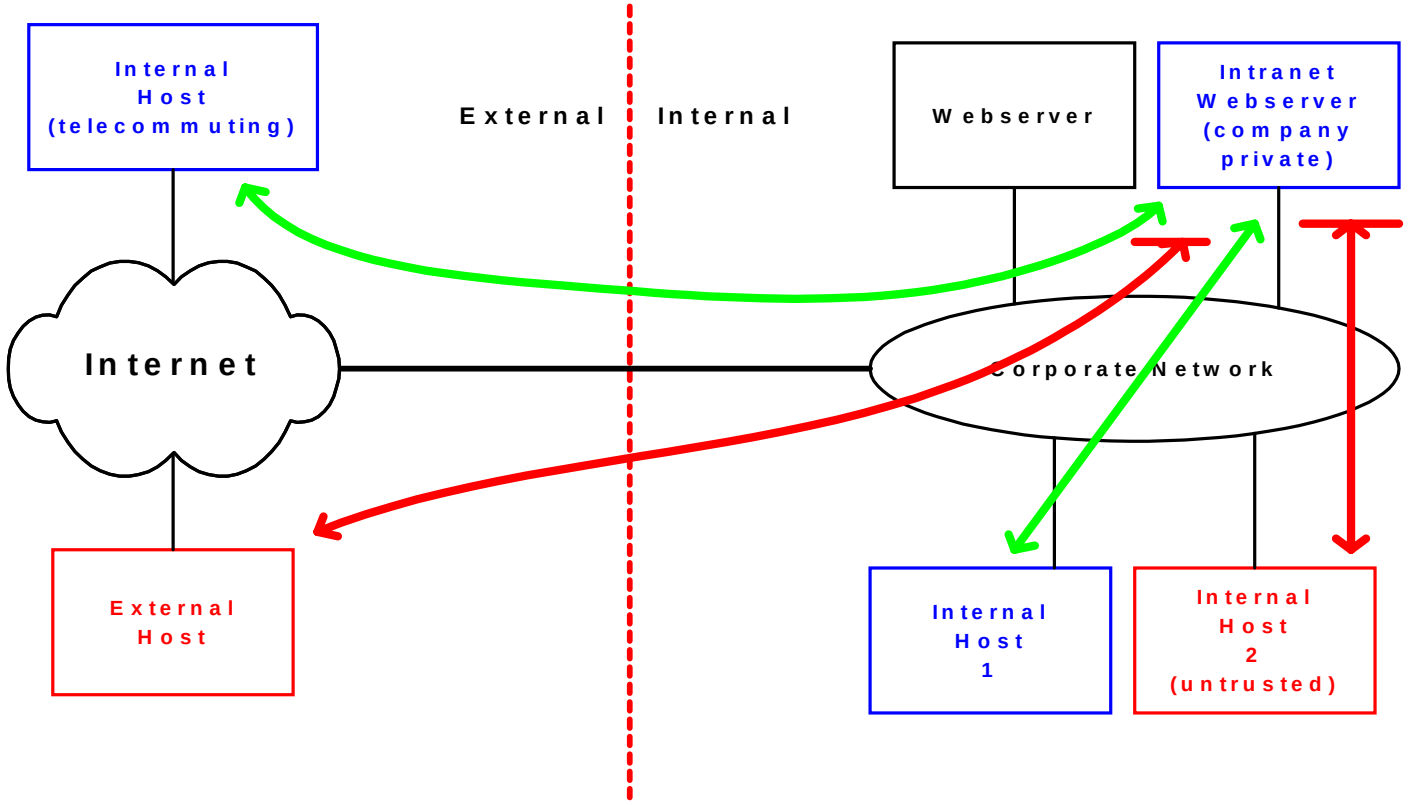
Distributed Firewall Example



Distributed Firewall Example to web server



Distributed Firewall Example to intranet



Distributed Firewall Implementation

- Language to express policies and resolving requests (KeyNote system)
- Mechanisms to distribute security policies (web server)
- Mechanism that applies security policy to incoming packet (Policy daemon and kernel updates)

KeyNote

- A language to describe security policies (RFC 2704)
- Fields in an “assertion”:
  - *KeyNote Version* – Must be first field, if present
  - *Authorizer* – Mandatory field, identifies the issuer of the assertion
  - *Comment*
  - *Conditions* – The conditions under which the Authorizer trusts the Licensee
  - *Licensees* – Identifies the authorized, should be public key, but can be IP address
  - *Local-Constants* – Similar to environment variable
  - *Signature* – Must be last, if present
- All field names are case-insensitive
- Blank lines not permitted within an assertion

KeyNote Policies and Credentials

- Policies and Credentials have same basic syntax
- Policies are “local”
- Credentials are “delegated” and MUST be signed

KeyNote Example 1

```
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:986512a1" || "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"
```

KeyNote Example 2

```
KeyNote-Version: 2
Authorizer: “rsa-hex:1023abcd”
Licensee: “IP:158.130.6.141”
Conditions: (@remote_port < 1024 &&
            @local_port == 22 ) -> “true”;
Signature: “rsa-sha1-hex:bee11984”
```

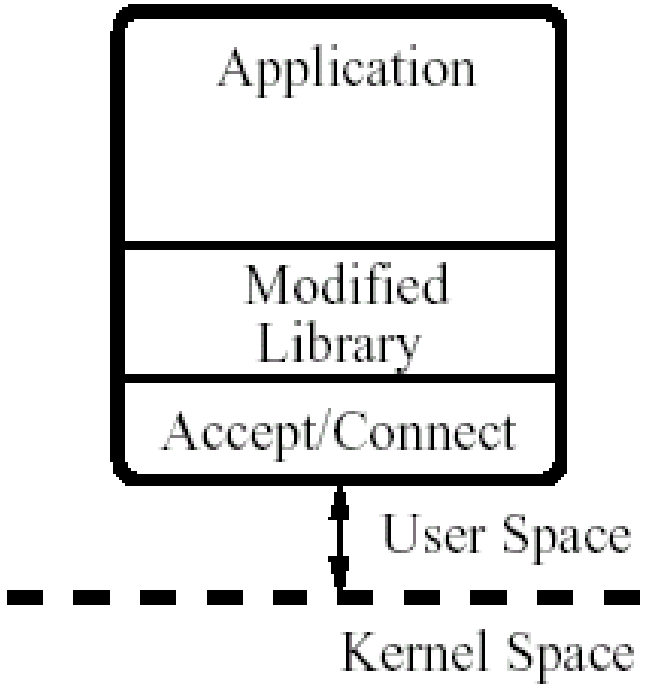
Note that this credential delegates to an IP address,

Distributed Firewall Implementation

- Not a complete solution, only a prototype
- Done on OpenBSD
- Filters done in kernel space
- Focused on TCP connections only
  - `connect` and `accept` calls
  - When a `connect` is issued, a “policy context” is created

User Space

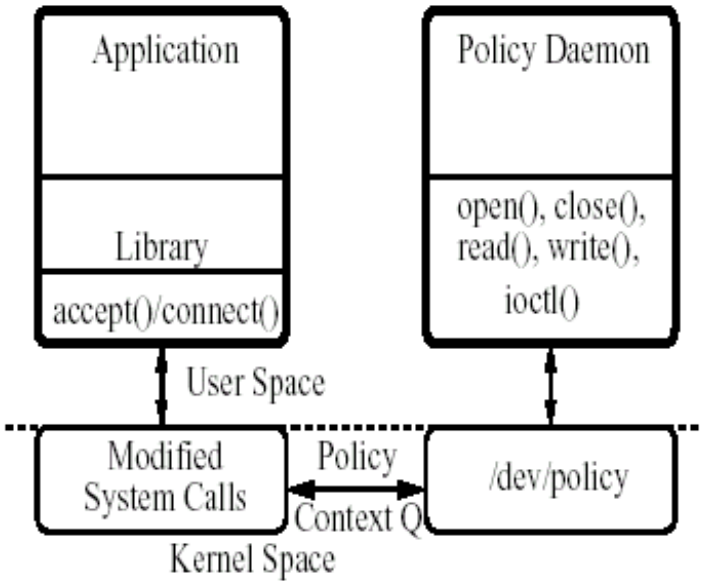
- This design was not chosen because of the difficulty in “forcing” an application to use the modified library
- For example, “telnetd”, “ftpd”



Policy Context

- Policy context contains all the information that the Policy Daemon will need to decide whether to allow or disallow a packet
- No limit to the kind of data that can be associated with the context
- For a `connect`, context will include ID of user that initiated the connection, the destination address and destination port.
- For an `accept`, context will include similar data to connect, except that the source address and source port are also included

Implementation Design



Policy Daemon

- User level process that makes all the decisions based on policies
- Initial policies are read from a file
- Current implementation allows changes to policies but changes only affect “new” connections
- A host that does not run this daemon is not part of the “distributed firewall”

Policy Device

- /dev/policy – pseudo device driver
- Communication path between the Policy Daemon and the “modified” kernel
- Supports standard operations: `open`, `close`, `read`, `write`, `ioctl`
- Independent of type of application

Example of Connection to a Distributed Firewall

- local host security policy:

```
KeyNote-Version: 2
Authorizer: “POLICY”
Licensees: ADMINISTRATIVE_KEY
```

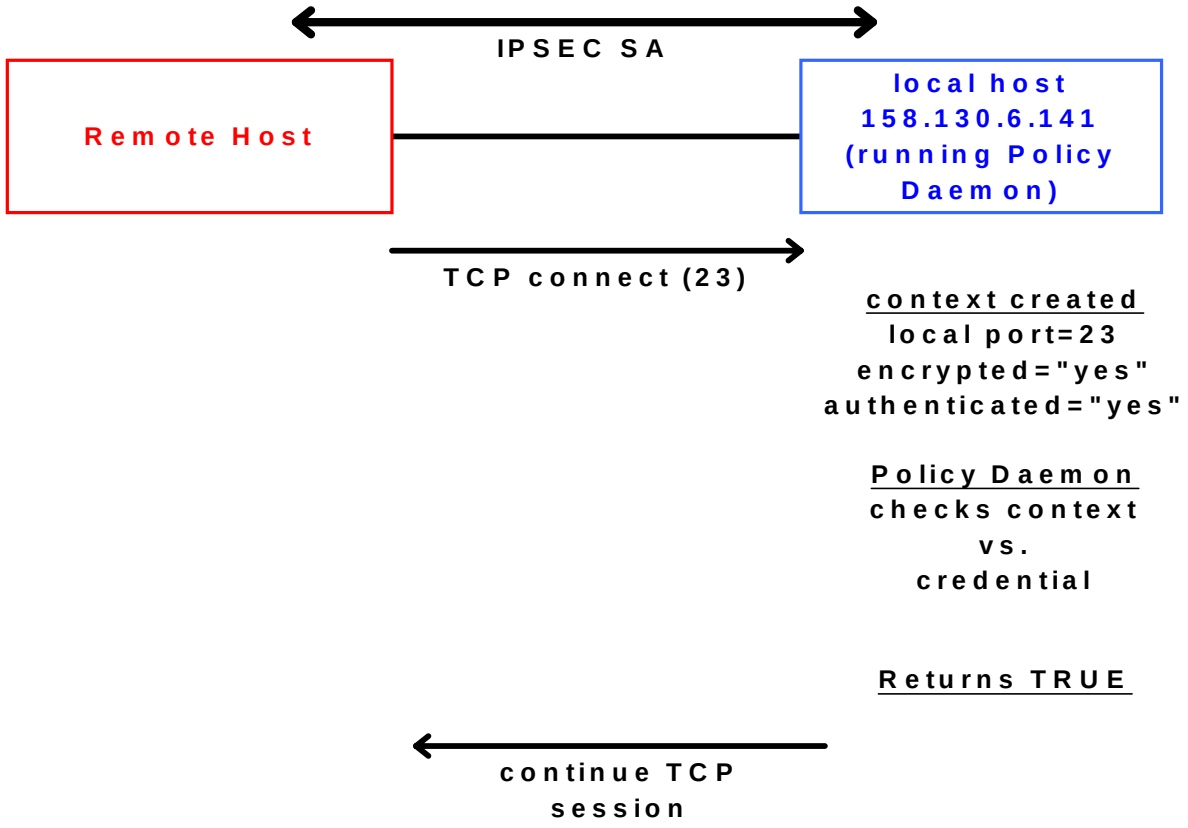
- Assumes an IPSEC SA between hosts

Example of Connection to a Distributed Firewall

- Credential provided to local host during IKE exchange

```
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 of Connection to a Distributed Firewall



Conclusions

- Distributed firewalls allows the network security policy to remain the control of the system administrators
- Insiders may no longer be unconditionally treated as “trusted”
- Does not completely eliminate the need for traditional firewalls
- More research is needed in this area to determine robustness, efficiency, and scalability

Future Work

- High quality administration tools NEED to exist for distributed firewalls to be accepted
- Allow per-packet scanning as opposed to per-connection scanning
- Policy updating and revocation
- Credential discovery

Questions