Architecture and Applications for a Distributed Embedded Firewall

Charles Payne
Tom Markham
Secure Computing Corporation
{*cpayne,markham*}*@securecomputing.com*

Abstract

The distributed firewall *is an important new line of network defense. It provides fine-grained access control to augment the protections afforded by the traditional perimeter firewall. To be effective, though, a distributed firewall must satisfy two critical requirements. First, it must embrace a protection model that acknowledges that everything behind the firewall may not be trustworthy. The malicious insider with unobstructed access the network can still mount limited attacks. Second, the firewall must be tamper-resistant. Any firewall that executes on the same untrusted operating system that it is charged to protect begs the question: who is protecting whom? This paper presents a new distributed,* embedded *firewall that satisfies both requirements. The firewall filters Internet Protocol traffic to* and from *the host. The firewall is tamper-resistant because it is independent of the host's operating system. It is implemented on the host's network interface card and managed by a protected, central policy server located elsewhere on the network. This paper describes the firewall's architecture and associated assurance claims and discusses unique applications for it.*

1. Introduction

Traditional perimeter firewalls are a critical component of network defense, but they should not be considered the *only* line of defense. First, their protection is too coarse. This leaves the firewall helpless against the *malicious insider*, who operates freely within the firewall's security perimeter. Second, it is costly to extend their protections to mobile users, because the firewall's security perimeter is determined somewhat artificially by the firewall's location in the network topology. For effective network defense, we must augment perimeter firewalls with more fine-grained access controls.

Bellovin [1] argued that a *distributed firewall* provides the fine-grained protection that is needed. In this solution, a firewall is placed at each host in the network, and all firewalls are managed as a single entity. That is, centralized management is coupled with distributed enforcement. Distributed firewalls contain the malicious insider because the security perimeter is drawn around *each host*. Because the perimeter is no longer defined by network topology, the distributed firewall is an ideal solution for mobile users, telecommuters and business-to-business extranets. Also, since distributed firewall policy is expressed in terms of network endpoints, changes to network topology have little if any impact on policy management. Ioannidis, Keromytis, Bellovin and Smith [3] described a prototype distributed firewall for OpenBSD hosts.

The distributed firewall falls short, however, if it assumes that all users on the local host are trustworthy. If these users are trusted to access the network freely, limited attacks such as network sniffing, host address spoofing and denial of service are still possible. To *effectively* contain the malicious insider, distributed firewalls must embrace a stronger protection model that acknowledges that users on the host may not be trustworthy. In other words, in addition to protecting the host from a malicious network, the firewall must protect the network from a malicious host. This requirement becomes more significant when we recognize that many insider attacks are not mounted intentionally. Worms like Code Red and NIMDA, for example, can turn loyal users into *unwitting insiders* [4].

The distributed firewall also falls short if it executes on an untrusted operating system. So-called *personal firewalls* suffer this fate. These software-based solutions fail to satisfy a cornerstone requirement for firewalls: tamper-resistance. Personal firewalls are relatively easy to disable via a network-based attack [7]. Like the emperor's dressmaker, they can leave the host — and by consequence the network — a bit exposed.

This paper describes a new distributed *embedded* firewall called *EFW* that embraces the stronger protection model and that is tamper-resistant. EFW filters Internet Protocol (IP) traffic to *and from* the host. EFW is tamper-resistant because it is independent of the host's operating system. In-

stead, the firewall is implemented on the host's network interface card (NIC) and managed by a central, protected policy server elsewhere on the network. EFW is implemented on a commodity NIC and scales easily to thousands of hosts.

The paper focuses on EFW's architecture and how it can support interesting security applications. Section 2 defines EFW's security and non-security objectives. Sections 3 and 4 illustrate the component and management architectures, respectively, that result from these objectives.

Distributed firewalls like EFW offer new opportunities in security policy enforcement. Section 5 enumerates several novel applications that we have identified for EFW.

Finally, EFW's genesis occurred in DARPA-sponsored research from the late 1990's. Since then, we have investigated its use in many problem domains. Section 6 offers a glimpse at EFW's future directions.

The remainder of this section considers related work.

1.1. Related Work

While EFW's goals and objectives closely resemble those of Bellovin [1], the two efforts proceeded independently and resulted in very different implementations (Bellovin's implementation is described in [3]). Bellovin noted that "for more stringent protections, the policy enforcement can be incorporated into a tamper-resistant network card" [1, Section 7.5], but he chose to implement his distributed firewall with kernel extensions, a user level daemon and a new device driver. Besides offering a simpler development path for a prototype, this strategy enabled the firewall to handle application-level policies. EFW, on the other hand, focuses on IP packet filtering because of the limited processing power available on the NIC.

Nessett and Humenn [5] proposed a novel multilayer firewall that can be managed centrally. Nessett's firewall includes all of the devices, such as perimeter firewalls, switches and routers, that currently perform filtering in the network. This work illustrates the pitfalls that can be encountered when firewall policy management is inextricably bound to network topology management. Bellovin [1] and Markham [4] advocate breaking this bond. Nessett and Humenn's results also underscore the importance of creating multiple layers (e.g., distributed firewalls and perimeter firewalls) in an overall network defense strategy.

2. Objectives for EFW

We divide the objectives for EFW into two camps: security-related and non-security-related.

2.1. Security Objectives

Figure 1 illustrates an EFW NIC on a protected host, the EFW policy server (also protected by an EFW NIC), and the communication paths between them. To illustrate our high-level design strategy, we state the security objectives for EFW in the form recommended by [6]. Assertions that EFW must satisfy are expressed as *claims*. Following the statement of each claim are zero or more assumptions upon which the claim relies. Validated assumptions are represented as claims elsewhere in this section and are so referenced. Unvalidated assumptions represent potential vulnerabilities for EFW that must be validated by other means (procedural controls, physical security, and so forth).

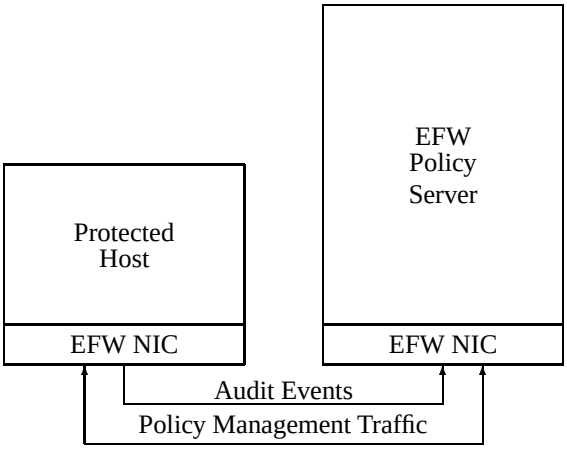

Figure 1. EFW NIC and Policy Server

The top-level claim is that EFW performs its function correctly.

Claim 1 *EFW blocks unapproved IP traffic to and from the host, which assumes*

- *EFW is configured properly*
- *EFW is non-bypassable [Claim 2]*

The first assumption captures the importance of *strength in policy*, while the second assumption captures the importance of *strength in mechanism*. The first assumption is validated on a case-by-case basis. Essentially we must ensure that the policy enforced by EFW is appropriate for the operational environment and its security threats. We will not consider this requirement further except to describe, in Section 4, the tools that EFW provides for policy management.

Claim 2 *EFW is non-bypassable, which assumes*

- *The host can communicate only through EFW-enabled NICs*

- *EFW is tamper-resistant to host-based attack [Claim 3]*
- *EFW is tamper-resistant to network-based attack [Claim 4]*

The first assumption is not trivial to achieve, and currently EFW offers no technical means to validate it. This means that we cannot, for example, stop the user from swapping out the EFW NIC for a non-EFW NIC. However, technical measures do exist in the EFW policy server to *detect* such activity. Fortunately, this potential vulnerability is temporary. Section 6 describes a technology that will prevent the host from accessing network resources *unless* it communicates through an EFW NIC.

Claim 3 *EFW is tamper-resistant to host-based attacks, which assumes*

- *The EFW NIC hardware is protected from direct manipulation*
- *Only the EFW policy server can disable an EFW NIC [Claim 5]*
- *Only the EFW policy server can download new policy [Claim 6]*
- *Attackers cannot masquerade as the EFW policy server [Claim 8]*

Claim 4 *EFW is tamper-resistant to network-based attacks, which assumes*

- *Only the EFW policy server can disable an EFW NIC [Claim 5]*
- *Only the EFW policy server can download new policy [Claim 6]*
- *Attackers cannot masquerade as the EFW policy server [Claim 8]*

The first assumption in Claim 3 can be validated by restricting the hardware interfaces. Newer generations of the 3CR990 NICs take steps in that direction by combining more functions onto fewer chips. The remaining assumptions for Claim 3 also apply for Claim 4. That is not a coincidence. While we typically imagine the EFW NIC as being managed remotely, the EFW policy server can protect itself with an EFW NIC, which it will manage locally. The protection mechanisms implemented on the EFW NIC do not distinguish whether the policy server is local or remote, and the host enjoys no privileged access to the EFW NIC.

Similarly, the next two claims have identical supporting assumptions.

Claim 5 *Only the EFW policy server can disable an EFW NIC, which assumes*

- *The operation is available only by a command to the EFW NIC API*
- *The command is accepted only from the EFW policy server [Claim 7]*
- *Only authorized users can access the EFW policy server*

Claim 6 *Only the EFW policy server can download new policy to an EFW NIC, which assumes*

- *The operation is available only by a command to the EFW NIC API*
- *The command is accepted only from the EFW policy server [Claim 7]*
- *Only authorized users can access the EFW policy server*

The first supporting assumption for Claims 5 and 6 is validated by the EFW implementation. The third supporting assumption can be validated by procedural controls and physical security. The remaining assumption is validated by a combination of technology, procedural and physical security controls, as described below.

Claim 7 *The command is accepted only from the EFW policy server, which assumes*

- *All policy server/NIC communications is authenticated by 3DES [Claim 9]*
- *Only the EFW policy server and the EFW NIC possess the cryptographic key [Claim 10]*

The remaining assumption from Claims 3 and 4 is validated by the same controls.

Claim 8 *Attackers cannot masquerade as the EFW policy server, which assumes*

- *All policy server/NIC communication is authenticated by 3DES [Claim 9]*
- *Only the EFW policy server and the EFW NIC possess the cryptographic key [Claim 10]*

The next claim defines the technology controls.

Claim 9 *All policy server/NIC communication is authenticated by 3DES, which assumes*

- *The work factor to break 3DES is too high*

The last claim defines the procedural and physical security controls.

Claim 10 *Only the EFW policy server and the EFW NIC possess the cryptographic key, which assumes*

- *Only authorized users can access the EFW policy server*
- *EFW crypto keys are protected from compromise*

2.2. Other Objectives

While most of the research behind EFW was funded by the US Department of Defense (DoD), the DoD relies increasingly on commercial-off-the-shelf solutions, so the needs of DoD and the commercial marketplace are not dissimilar. As a result, we also considered commercial viability and commercial acceptance throughout this effort. In addition to being secure, we determined that EFW needed to be cost-effective, scalable and friendly to manage.

Cost-effective. The constraints of implementing on a NIC prompted the motto: "fast, simple and cheap". For performance, a NIC has a tight processing loop, and our solution had to fit within those bounds. A NIC also has limited memory, so complex processing is performed elsewhere (e.g., on the EFW policy server). Lastly, the 3CR990 NIC is relatively inexpensive, and we did not want to alter that fact. These NICs are already widely deployed, so modifications to the existing hardware and its drivers were avoided at all costs. We confined our modifications to the NIC's firmware.

Scalable. To facilitate commercial acceptance, it must be possible for administrators to introduce EFW as little or as much as they like. Initially, some administrators may prefer to protect only a few critical servers; others may immediately deploy EFW to every client desktop along with a policy that enforces "good network hygiene". The differences between a large deployment (thousands of NICs) versus a small one are minimized in the eyes of the administrator through the use of management abstractions (explained in Section 4). We also adopted a master/slave architecture between the policy server and its NICs.

Friendly to manage. To make EFW friendly to manage, we created several administration tools, including a policy editor, a EFW device (NIC) manager, and an audit logger and event viewer. These tools rely on several abstractions to reduce their complexity. Our objective was to make EFW invisible to the end user and to incorporate familiar management paradigms for the administrator.

3. Implementing EFW

The high-level architecture for EFW is illustrated in Figure 2. The protected host may be a client workstation, a server, or any other device that supports the NIC. The policy server should be installed on a dedicated host and protected by its own EFW NIC. The following sections describe the components on each platform in greater detail.

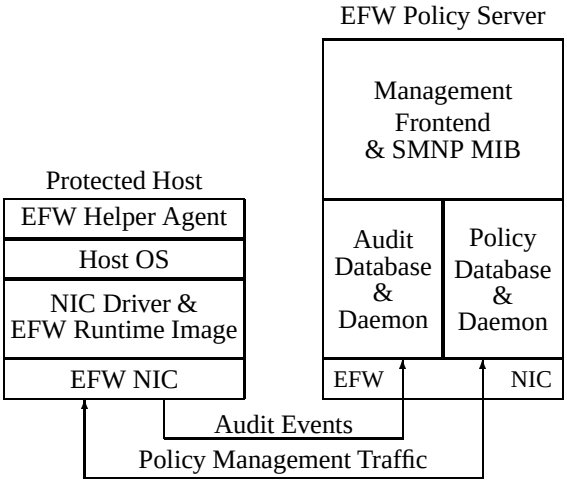

Figure 2. EFW Architecture

3.1. EFW Components on the Protected Host

Three components reside on each protected host: the EFW-enhanced NIC, the NIC's driver and runtime image, and a non-security-critical helper agent.

EFW-enhanced NIC. The most important component on the protected host is the NIC and its EFW-enhanced firmware. EFW is based on the 3Com 3CR990 family of NICs. We selected these NICs for several reasons. First, they have an on-board processor and memory, which allows the NIC to operate independently of the host operating system. Second, they contain an on-board cryptographic engine. This feature was included to support Windows 2000 IPSEC offloads, but EFW also leverages the crypto engine to provide secure communications with the policy server. Finally, these NICs are relatively inexpensive and widely available.

Flashed onto the NIC during the EFW install, the EFW-enhanced firmware contains the packet filtering engine and the management interface for the EFW policy server. The packet filter can accept or reject packets according to the standard parameters (source and destination address, source and destination port range, IP protocol, packet direction, etc.) as well as the value of the TCP SYN flag (used for

connection initiation) and the presence of IP options. It can also accept or reject fragmented packets and non-IP packets. Each filter rule can be configured to generate an audit event. The management interface handles policy downloads from the policy server and transmits audit events to the audit server. It is also responsible for managing the secure channel with the policy server.

Driver and runtime image. The driver installed for a EFW NIC is the unmodified commercial driver. Like similar products, this NIC relies on its driver upon each host reboot to download its runtime image into the firmware. To ensure that a host remains protected, the EFW NIC stores enough information in non-volatile memory to verify the integrity of its runtime image. Once a NIC is configured for EFW, it cannot be disabled except by performing the appropriate action on the policy server. In other words, the EFW NIC will become inoperable if its runtime image fails the integrity check.

Helper agent. The NIC must know its IP address in order to enforce policy. In a DHCP environment, it will need the host's assistance to determine that address. A small helper agent in user space performs this function. The helper agent also sends regular heartbeats to the policy server to help the policy server detect NICs that may not be functioning. Like all other communications with the policy server, the heartbeat is encrypted by the EFW NIC. If a malicious user were to replace the EFW NIC with a vanilla NIC, the heartbeats for that EFW NIC would effectively stop, raising the suspicions of the EFW administrator. The EFW NIC does not rely on the helper agent for continued operation and will continue to enforce policy even if the helper agent crashes or is removed.

3.2. EFW Policy Server Components

The EFW policy server is composed of three main components:

1. the *management component*, including the graphical user interface (GUI), the SNMP management information base (MIB) and the controller frontend,
2. the *policy component*, including the policy daemon and the policy database, and
3. the *audit component*, including the audit daemon and the audit database.

Management component. The management component is described more fully in Section 4. Its main purpose is to provide the administrator with the tools to create, view and distribute policies to each EFW NIC. It also includes an audit browser to review event logs. The MIB will support future network management applications.

Policy component. The policy component takes the policies defined using the management component and compiles them into filter rules for each NIC. This component ensures that NICs enforce the policy to which they are assigned. When a NIC's host is rebooted, the NIC requests the current policy from the policy server. If the policy server does not respond, the NIC "falls back" to enforce a simpler policy. Currently the choices are: allow all traffic, allow all traffic but prevent network sniffing, or deny all traffic. If a policy is modified, the policy component automatically pushes the updated policy to the affected NICs. NICs that are off-line during the policy push receive the policy once they return on-line. The heartbeat generated by the host's helper agent informs the policy component of the policy that the NIC is enforcing.

Audit component. The audit component receives audit events from each NIC and stores them in a database for browsing and searching by the management component. Audit logs can also be exported to third-party tools for additional analysis. As the arrows in Figure 2 imply, policy updates from the policy component to the NIC are acknowledged by the NIC; however, the audit component does not acknowledge audit events generated by the NIC.

4. Managing EFW

EFW provides many useful abstractions to help the administrator define and manage policies. This section describes those abstractions and discusses the challenges and opportunities of managing distributed firewalls.

Abstractions. EFW divides protected hosts into *policy domains*. A policy server can manage only one policy domain, although there may be multiple policy servers for each domain. A policy domain might encompass an entire organization or perhaps just one or two divisions within that organization.

Within each policy domain, NICs are grouped by function into *device sets*. There might be one device set for managers, another for the finance staff, etc. Device sets reduce complexity by grouping together the NICs that are likely to be assigned the same policy. So while there might be thousands of NICs in a policy domain, there may only be a dozen or so device sets. Each device set is assigned a single policy, although a particular policy may be assigned to multiple device sets.

Policies are composed of *policy attributes* and *rules*. Policy attributes represent facts that apply across all rules

of the policy. For example, "the host is not allowed to spoof its IP address", or "fragmented packets are not permitted". EFW rules are similar to the packet filter rules found on other firewalls. For convenience, rules can be grouped into *rule sets*. If any rule in the rule set is modified, the changes propagate to all policies that include the rule set.

EFW also supports *audit* and *test mode*. Audit can be set for an entire policy or just for an individual rule. Test mode works with audit to help the administrator understand the effects of a policy or a single rule before it is actually enforced. For example, a rule that is in test mode will generate audit events each time a packet matches it, but the action associated with that rule (allow or deny) will be ignored.

Audit is also very useful for "discovering" policy. For example, to identify the network services that a particular host requires to boot up and log on users to the net, we can push an "allow but audit" policy to its EFW NIC. Then we reboot the host and watch the audit logs. A network monitor would perform a similar function.

Challenges. EFW is not immune to the policy management challenges that face other packet filters, and the limited resources of the NIC make overcoming these challenges even more difficult. For example, port mapping protocols, i.e., protocols that start on a well-known port then negotiate a higher, random port to complete the session, require the firewall to maintain some state about the session. Protocols that use a well-known control port and a random data port (e.g., FTP and some streaming media protocols) are similarly challenging. We are examining alternatives for solving this problem for EFW. Fortunately, the challenge exists only if we need to specifically allow these protocols while denying everything else. If we want to deny these protocols, we can deny the connection to their well-known ports.

Opportunities. Managing policy for a distributed firewall like EFW is not simply a matter of moving the perimeter firewall's policy to each endpoint EFW device. Consider the following policy:

> Allow HTTP requests from a specific client to a specific web server.

On a traditional firewall, this policy might be stated as a single rule (see Table 1), where a rule is stated in the form *(action, protocol, port, source, destination)*. We assume that traffic is permitted in both directions.

(Allow, TCP, 80, client, web server

Table 1. Traditional Firewall

Placing this rule on both the EFW for the client and the EFW for the web server would be redundant, and if it was the only rule enforced by either EFW, it would probably be overly restrictive. More likely, the policy writer would choose to distribute the policy between the two devices, such as illustrated in Table 2. However, the same effect could be achieved by distributing the policy as in Table 3. In both tables, the web server is restricted to processing HTTP requests. However in Table 2, the client may make other requests, while in Table 3, the client is restricted to HTTP requests.

for host client
(Allow, *, *, client, *)
for host web server
(Allow, TCP, 80, *, web server)

Table 2. EFW — Option 1

for host client
(Allow, TCP, 80, client, *)
for host web server
(Allow, TCP, 80, *, web server)

Table 3. EFW — Option 2

Tables 2 and 3 express different policies from each other and from Table 1; however, the differences are only evident when we express the policies for EFW. The traditional firewall policy did not specify the behavior of the client and the web server beyond HTTP requests flowing through the perimeter firewall. For a particular site, these distinctions may be important, and EFW helps us to make them. EFW lets the administrator state policies far more precisely.

5. EFW Applications

While EFW can certainly handle applications conceived for traditional, packet-filtering firewalls, its real power lies in applications that are either not possible or not feasible using traditional firewalls. This section describes several useful applications that we have encountered. Each application forms a building block that can be used to construct even more interesting applications.

No sniffing, no spoofing. One of the most significant applications for EFW is its ability to enforce good network hygiene. In general, a host should not be able to sniff other network traffic or spoof its IP address to other hosts. Many network attacks rely on one or both behaviors. Distributed denial of service attacks, for example, direct zombies to flood the victim host with spoofed service requests. EFW can prevent the NIC's untrusted driver from placing the NIC in promiscuous mode, and it can also prevent any packet

from leaving the host that is not tagged with a valid IP address for that host.

Lock down the host. One of the biggest problems for IT personnel is keeping up with the security patches that must be installed. Often these patches are for services that are installed by default when the operating system is installed. Sometimes the services are network services that the user should not be invoking anyway. Rather than manually reconfiguring each host to disable the service, EFW can prevent the service from being available.

Another problem is users who configure their hosts in violation of the organization's security policy. For example, most network administrators prefer that users share files using an IT-maintained resource. However, a user can easily configure the typical PC to share files from the local disk. EFW can prevent this behavior by preventing file access requests from reaching the host.

Servers are not clients. Dedicated servers should not perform certain functions normally reserved for clients, such as sending email, making web requests and so on. The NIMDA worm, for example, relies on this behavior to propagate. For TCP-based services, the easiest defense is to prevent the host from initiating TCP connections to other hosts. EFW can prevent unauthorized, outgoing TCP connection initiation requests from ever reaching the network.

Clients are not servers. Similarly, client hosts should not respond to service requests from other hosts. For TCP-based services, EFW can prevent unauthorized, incoming TCP connection initiation requests from ever reaching the host.

Stay in your own backyard. With the exceptions of web and certain related traffic (FTP, streaming media, etc.), client hosts obtain most network services from dedicated LAN (local area network) servers. For example, it is typically not necessary for a user to access any DNS or SMTP server other than the one assigned by IT. To accomplish this goal, EFW can allow limited "external" requests, then restrict all other traffic to the local subnet.

Don't talk to strangers. If a particular network service, e.g., DNS, should be provided only by a specific server or group of servers, then EFW can restrict access to that service on only that server or group of servers.

Emergency rule set. This very useful application utilizes the rule set feature of the EFW policy server. Rule sets are included in policies by reference, not by value, so a change in the rule set propagates to all affected policies. Using this capability, an administrator can define an *emergency* rule set to be included in all policies. If a network attack is detected that requires a particular service port, the administrator can add a rule to the emergency rule set denying that port. With a click of a single button, the administrator can distribute this new restriction to all EFW NICs in the EFW policy domain.

Shared server. Business-to-business communications require an infrastructure for sharing information. Extranets are the common solution, but extranets are expensive to implement. EFW enables a lightweight, cheaper alternative: the *shared server*.

A single host with two EFW NICs is placed where it is accessible by both organizations. The organization that hosts the server controls both EFW NICs. We assume that both organizations may have administrator privileges to the server. The EFW NIC that is attached to the controlling organization's LAN prevents the shared server from initiating unauthorized communication on the LAN and sniffing traffic on the LAN. The EFW NIC that is attached to the Internet allows only protected communications with the business partner. The business partner can enter and access the shared server, but it is unable to exit out the "other side".

6. Future Work

As we gain more experience with EFW, we envision applications will require features not yet present in the architecture. For example, through our DARPA-sponsored research programs, we are currently investigating tie-ins to intrusion detection and response systems and using the EFW NICs to provide load sharing within server clusters.

Another important area of investigation is a technology we call *virtual private groups* (VPG) [2]. Like a virtual private network (VPN), a VPG establishes a community of interest that is not restricted by network topology. Unlike the VPN, however, which establishes only pairwise relationships between the communicating entities, the VPG establishes group-wide relationships. The VPG architecture significantly simplifies key management for hosts within the group and makes management of secure group communications practical. When it is available, VPG technology will enable organizations to quickly set up, use, and then tear down secure group communications for wireless LANs and collaboration tools.

The VPG technology will also be an important catalyst for ensuring that network communication occurs only through EFW NICs (see the first assumption under Claim 2 in Section 2.1). If all hosts belong to one or more VPGs, and if network services are available only to members of the appropriate VPG, then only EFW NICs will be able to

access network services. The attacker who attempts to access the network with an unsecured NIC will be completely thwarted.

7. Summary

We have described a distributed, embedded firewall called EFW that is implemented on the host's network interface card. In addition, we have discussed several useful and unique applications for EFW. EFW can be used to lock down critical assets, such as corporate web servers, databases and administrative workstations, and it can be used to lock down critical services, such as DHCP, DNS and so forth. It lets the administrator easily control unnecessary capabilities on the network. Finally, EFW demonstrates that finer-grained network access control is possible and practical. Together with the perimeter firewall, it forms a strong line of network defense.

8. Acknowledgments

The authors are grateful for the financial support of the US Defense Advanced Research Projects Agency. This paper reflects work performed under the Releasable Data Products Framework program (Contract no. F30602-99-C-0125, administered by the Air Force Research Laboratory) and the Autonomic Distributed Firewall program (Contract no. N66001-00-C-8031, administered by the Space and Naval Warfare Systems Center). The authors also wish to thank the anonymous reviewers for their helpful and insightful comments.

References

[1] S. M. Bellovin. Distributed firewalls. *;login:*, pages 37–39, November 1999.

[2] M. Carney, B. Hanzlik, and T. Markham. Virtual private groups. In *Network and Distributed System Security Symposium*, February 2002. Submitted for publication.

[3] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In *7th ACM Conference on Computer and Communications Security*, Athens, GREECE, November 2000. ACM.

[4] T. Markham and C. Payne. Security at the network edge: A distributed firewall architecture. In *DISCEX II*, Anaheim, CA, June 2001. DARPA, IEEE.

[5] D. Nessett and P. Humeen. The multilayer firewall. In *Network and Distributed System Security Symposium*, March 1998.

[6] C. N. Payne, J. N. Froscher, and C. E. Landwehr. Toward a comprehensive INFOSEC certification methodology. In *Proceedings of the 16th National Computer Security Conference*, pages 165–172, Baltimore, MD, September 1993. NIST/NSA.

[7] A. White. New trojan disables firewall defences. *Network News*, May 2001.