Emulating an Embedded Firewall

Clifford Neuman, Deepak Dayama, and Arun Viswanathan
University of Southern California

Abstract

The Adventium Labs Embedded Distributed Firewall provides a simple interface for securely managing approved network flows between computers on a network. A "conversation" manager provides a simple interface for managing flows, defining the connections authorized between nodes on a network. These policies are enforced in hardware embedded in the network interface card of each computer. The policies are managed to create groups of communicating machines and services and to exclude undesired traffic.

This paper describes the emulation of the Adventium Labs distributed embedded firewall, using an additional node associated with each user node emulated on the DETER testbed. We provide observations on our implementation and current experiments, and discuss how the emulation can be used by other experimenters.

1 Introduction

The Adventium Labs Embedded Distributed Firewall [3] centrally manages network communication between computers on a network. Policy is generated on a management node using a conversation manager [4] and transmitted to a hardware firewall on the network interface card of each managed computer. Such firewalls are more resistant to compromise from malicious code on the host computer itself, and central management allows communication to be enabled for virtual groups of machines sharing the same physical network with other groups.

Responding to certain kinds of attacks, such as denial of service (DoS), may require dynamic changes to certain policies to block traffic that was once allowed, yet policy changes must be propagated across the network, and policy updates might themselves compete for bandwidth with the DoS flows themselves.

We were interested in understanding the interaction between the embedded firewall policy updates, and potential denial of service flows that policy updates are intended to mitigate. We were also interested in developing a capability to model such embedded firewalls on the DETER testbed [1], even without the actual hardware firewalls in NIC cards, and for this capability to be available for other experimenters.

2 Distributed and Embedded Firewalls

Ioannidis, Keromytis, Bellovin and Smith [2] introduced the concept of a distributed software firewall in 2000 to address the classic problem of providing continued protection to end systems even once an outer perimeter firewall has been breached. A distributed firewall is implemented on each computer in a network, but managed centrally to enforce an organization's relevant policies for network communication.

Subsequently, Prevelakis and Keromytis [5] introduced an embedded firewall (EFW), a host-based firewall implemented in separate hardware, placed between a host and the rest of a network, addressing software compatibility problems and improving resistance to compromise from within an infected system. 3Com Corporation and others offers embedded firewalls that resides on network interface cards, and the 3Com hardware is used in the Adventium Embedded Distributed firewall solution.

A common characteristic of most distributed firewalls is that updates to policy are communicated through the same network whose network flows are controlled by the managed firewalls.

3 Modeling the Adventium Firewall

The DETER testbed provides researchers with an environment for conducting repeatable computer security experiments [1]. The testbed was built using Utah's EMULAB [7], and has been configured and extended to provide stronger assurance of isolation

Clifford Neuman, Deepak Dayama, Arun Viswanathan. Emulating an Embedded Firewall.
In Proceedings of the DETER Community Workshop on Cyber-Security and Test, August 2007, Boston.
http://clifford.neuman.name/publications/2007/200708-usecdw-emulating-embedded-firewall/

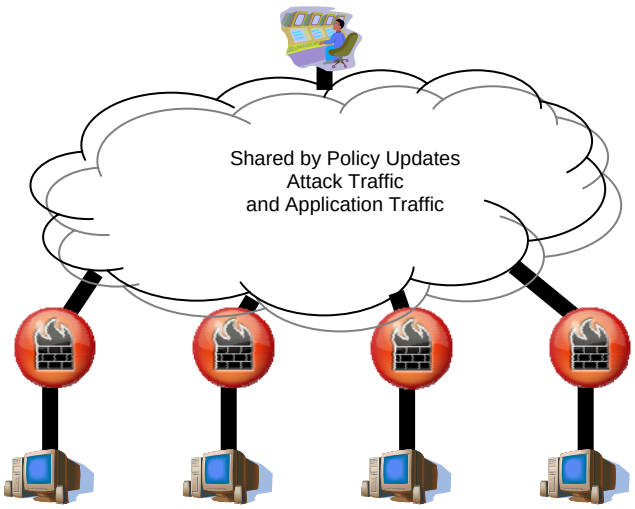

Figure 1. Policy Dissemination and Attack Topology

and containment. At present, no nodes or network interface cards within the DETER testbed are configured with embedded firewalls. We wanted to model network configurations with such firewalls, and to provide that capability for other DETER users studying the effect of alternate network topologies on security.

By using a second testbed node associated with each application node, we were able to use the DETER testbed to emulate the function of the embedded firewalls. We developed a tool to translate the policy language used by the Adventium conversation manager to iptables, which were then loaded on the "firewall" nodes to model the use of an embedded firewall. Since we were interested in studying the effect on policy updates, it was also necessary to model the management node on which the conversation manager disseminates updates, and to consider alternate topologies for the dissemination of firewall policies. Figure 1 shows the interconnection of firewall nodes and manager used in our study, relative to the interconnection of the production network, and the nodes protected by the firewalls.

4 Representing Rule Sets

Because of the hardware and base operating system configurations readily available on the DETER testbed, we chose to emulate the embedded firewall using netfilter [6] and to manage the local policy rules using iptables.

Netfilter is the successor to ipchains as a package for filtering network packets within the linux operating system. Iptables is a user space tool that enables the specification and creation of the filters applied by Netfilter. The Adventium Labs Embedded Distributed firewall policies are represented in a format that is more readily suited to identifying specific allowed communication among a set of nodes, and these policies needed to be converted to a representation more readily used by iptables before they can be loaded into an emulated embedded firewall node implemented with netfilter.

Representation of firewall rules

We were provided with an initial version of the rule set language used by the Adventium Labs conversation manager for representing permitted communication through their embedded firewall. In the initial version, EFW rules were retrieved and in most cases, the firewall policies were readily converted to equivalent iptables notation. The handling of group memberships however, required using the 'ipset' extensions to ipfilter. Ipset is a utility and set of kernel patches which allowed us to create groups with multiple addresses and ports which are to be matched to the same rule. These sets correspond to placeholders in the Adventium Labs firewall rules.

To use iptables to represent the Adventium EFW rules, we first create sets of ip addresses using ipsets which contain the member addresses and ports of the sources and destinations corresponding to the placeholders in the EFW rulesets. Our strategy was to look for the presence of either source or destination group membership for the packets, and direct them to user-defined ipchains which apply to those sets of addresses. In effect, we have one rule in the INPUT and OUTPUT chains of iptables for every permitted group in the EFW file. When a packet arrives at the firewall, the rules on the INPUT chain cause a check for membership of the packet source for all permitted groups. When a match is found, the packet is forwarded to a user defined chain in the iptables corresponding to the packet's group.

Figure 2 shows the traversal of a packet through iptables in our emulated embedded distributed firewall. Each user-defined chain corresponds to at least one rule name in the EFW rule-set. The remaining part of the packet, including destination

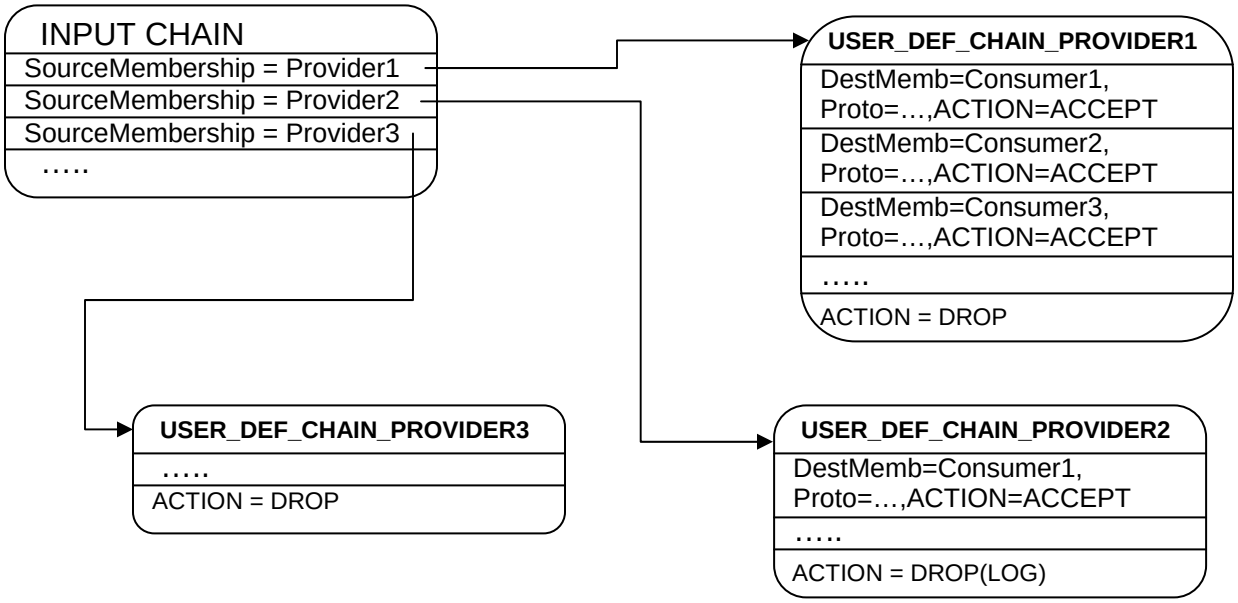

Figure 2. Traversal of packet through iptables

group membership and protocol are matched in the user-defined chain. When multiple destination groups match the same source address in EFW rules, two or more rules of the embedded firewall are applied by sequentially forwarding the packet to the next rule within the same chain when a match fails for earlier rules.

All rules within user-defined chains have an ACCEPT or DROP for a matching destination group (a set of host, port, protocol) and there is a DROP at the end of each chain in case a match is not found.

5 The experiments

In a relatively short time, we were able to successfully emulate the embedded firewall using a secondary Linux node with support for ipset and iptables, on an emulated network of on the order of 10 nodes. Our initial experiments used a static distribution of policies from the conversation manager, which were translated to iptables representation, which were then installed using a script. At this stage we were not using the required separate node as the conversation manager to communicate policies (or for that matter, changes to policies).

Once the emulation of the firewall was in place, the next step was to define the emulated network topologies on which the embedded firewalls were to run. Here it is necessary to consider special nodes, such as the computer on which the conversation manager runs, and to decide how policies would be disseminated to the embedded firewalls.

Figure one shows the representative topology, before introducing attack traffic. The conversation manager is resident on a single DETER node, and policies are pushed through the network to the DETER nodes that represent the EFW's. Unfortunately, because the implementation of the EFW's is different than the actual hardware NIC cards, experiments using the emulated EFW's will not be useful for understanding the performance of the EFW's themselves. The EFW extensions can be used, however, to understand the effects of the EFW's on network response under varying scenarios.

In particular we can vary attack traffic using tools already available on the DETER testbed, and we can model different strategies for disseminating policy changes from the conversation manager to the emulated embedded firewalls. If policies are to be dynamically updated in response to observed attacks, we are interested in knowing whether the attack traffic itself will prevent the necessary policy updates from being received and enforced by the EFWs.

We will consider approaches where policies are pushed to the EFW's nodes from the conversation manager, polled by the EFW's, as well as hierarchical approaches where changes are pushed to intermediate nodes and then redistributed. Finally, we can consider how the use of alternate initial EFW policies will affect the ability to disseminate updates in response to network attack.

Finally, we will want to know how accurately our EFW emulation models the production firewalls embedded on NIC cards. The size of the network configurations studied to date is not sufficient for us to draw conclusions regarding the fidelity of our emulations, nor do we have data from the NIC card implementations, but we need to look at the expected performance of the emulated EFW under varying circumstances of load and number of policies rules being implemented. Understanding such differences is important because delays in communicating network traffic across the emulated EFW would have an impact on experiments in DDoS response or malicious code propagation, where timing is critical.

These experiments are still ongoing and results will be available to DETER users as part of the documentation of the emulated embedded firewall package when it is available for use by other experimenters.

6 Leave-behind for experimenters

At present, setting up emulated embedded firewall experiments requires manual configuration of the topology for each experimental node, and installation of the scripts needed to accept policies from the conversation manager and install them through iptables. Those needing to use an embedded firewall on the DETER testbed would need to retrieve an archived experiment to use as a template, and modify it to meet their own experimental needs. This process needs to be simplified.

It is our intent to provide greater automation to help the DETER experimenter model embedded firewalls. The experimenter should be able to view a node with an embedded firewall as a single node, with special capabilities, rather than defining a topology with twice as many nodes, as modeled machines. Depending on the nature of the experiment, it might be possible to emulate the function of the embedded firewall using iptables on the host computer itself. Whether such a modeling of the embedded firewall would be accurate requires that we understand the benefit of embedding a firewall.

The first benefit is software independence, i.e. if the software on the end node to which the embedded firewall would have been attached can support Netfilter and iptables, then the end node can emulate the function of the embedded firewall without an auxiliary node. We can increase the likelihood of being able to emulate an embedded firewall on the end node itself if we develop alternate emulations for the most common system images in use on DETER.

The second benefit is one of isolation, i.e. the functioning of the embedded firewall itself is not subject to interference from other code running on the attached end node. With the exception of malware experiments, this benefit too is not necessary, and the emulation could be run on the end node itself.

To be fair, we should note that an experiment that does not require a separate node for running a firewall is really capable of running using a simple host-based firewall, a software commodity that is readily available. In such a case, an important aspect of this work is making it clear when the embedded firewall really is needed to properly model a DETER experiment, and to provide the management interfaces within an experiment that matches the conversation manager used by the Adventium Labs distributed embedded firewall.

It is our intent to make support for embedded firewalls an optional capability for configuring nodes within tools available to DETER experimenters.

7 Conclusion

Adventium Labs' conversation manager for the Distributed Embedded firewall provides a simpler interface for centrally defining policies enforced by a network of embedded firewalls. By providing tools to map the conversation manager's policy language to iptables notation, and by using a second testbed node associated with each user node, we are able to emulate such embedded firewalls on the DETER testbed without the special hardware used in production networks.

Our work to model the Adventium conversation manager and embedded firewall on the DETER testbed leaves behind code that is usable by other experimenters seeking to emulate network topologies that includes such hardware. For some experiments, in particular, those running common OS's and that do not involve malicious code, this support can be provided without the need to allocate a second node. Future support may allow embedded firewall configurations to be specified as options during the specification of the topology of an experiment.

8 Acknowledgements

This material is based on research that was supported by funding from the United States National Science Foundation (NSF) and the United States Department of Homeland Security (DHS) under contract numbers ANI-0335298 (DETER) and CNS-0454381 (DECCOR) and by the United States Air Force and the Department of Homeland Security HSARPA as a subcontract to Adventium Labs under contract number FA8750-05-C-0144. Opinions, findings, conclusions and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation, the United States Air Force, the Department of Homeland Security or Adventium Labs. Figures and descriptions are provided by the authors and are used with permission.

9 References

[1] Benzel, Terry, Robert Braden, Dongho Kim, Clifford Neuman, Anthony Joseph, Keith Sklower, Ron Ostrenga, and Stephen Schwab, *Design Deployment and Use of the DETER Testbed* In Proceedings of the DETER Community Workshop on Cyber-Security and Test, August 2007, Boston.

[2] Ioannidis S., A.D. Keromytis, S.M. Bellovin and J.M. Smith, Implementing a Distributed Firewall, *Proceedings of the ACM Conference on Computer and Communications Security (CCS) 2000*, pp. 190-199.

[3] Payne, Charles and Tom Markham. Architecture and Applications for a distributed embedded firewall. In 17th Annual Computer Security Applications Conference, December 2001.

[4] Payne, Charles. The Conversation Manager, Version 1.0. Adventium Labs report, October 2006.

[5] Vassilis Prevelakis, Angelos Keromytis, Designing an Embedded Firewall / VPN Gateway. *Proceedings of the International Network Conference* 2002, Plymouth, UK..

[6] Welte, H. The Netfilter Framework in Linux, Proceedings of the Linux Kongress 2000.

[7] White, B., J. Lepreau, L.Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated experimental environment for distributed systems and networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI02),* (Dec. 2002). Pp 255-270.