Extensible Network Configuration and Communication Framework

Todd Sproull and John Lockwood

Applied Research Laboratory
Department of Computer Science and Engineering:
Washington University in Saint Louis
1 Brookings Drive, Campus Box 1045
St. Louis, MO 63130 USA

http://www.arl.wustl.edu/arl/projects/fpx/reconfig.htm

Abstract. The effort to manage network security systems has increased in complexity over the past years. Network security for a company, university, or government agency can no longer be provided using a single Internet firewall or Intrusion Prevention System (IPS). Today, network administrators must deploy multiple intrusion detection and prevention nodes, traffic shapers, and firewalls in order to effectively protect their network. As the number of devices increases, maintaining a secure environment becomes difficult. This paper presents an infrastructure for control, configuration, and communication between heterogeneous network devices. The approach presented uses a Publish/Subscribe model built on top of a peer-to-peer overlay network in order to distribute information between network intrusion detection and prevention devices.

1 Introduction

Network administrators have become overwhelmed by the task of securing their networks against attacks on hosts in their network. End hosts are difficult to protect because they can be subverted via attacks against flaws in their operating system, trojan programs, and misconfiguration by users. Firewalls, Network Intrusion Detection Systems (NIDS) and Intrusion Prevention Systems (IPS) integrated within the network help protect against exploitation of such flaws. Many organizations also deploy traffic shapers to help organize bandwidth more fairly.

Today, network administrators spend much of their time patching or changing device configurations to prepare or recover from the latest computer worm or system exploit. The manual configuration and administration of each device contributes to down time of the network.

It is difficult to manage a group of firewalls, NIDS, IPS, and traffic shapers because today's different devices each user proprietary management infrastructure. Developing a common framework for these devices to communicate would greatly reduce the time associated with control and configuration of all systems

in the network. In this work, we propose a unified management network that exchanges information using an eXtensible Markup Language (XML).

The proposed infrastructure allows heterogenous nodes to communicate via XML messages sent over an adhoc, Peer-to-Peer (P2P) overlay network. Experimental results demonstrate the performance of the overlay forwarding high level security alerts.

2 System Architecture

2.1 Services

Nodes in the overlay network subscribes to services of interest. Nodes subscribe to the services by issuing discovery queries to the network. Nodes implementing services respond with a list of the services they offer as well as advertisements of other peers in the group.

Services include, but are not limited to: rule updates for intrusion detection and prevention, signatures distribution for viruses and trojan horses, anomaly detection and SPAM filters. In general, these services run in hardware or software in the network infrastructure and communicate through an overlay network.

2.2 Overlay Network

The P2P overlay network is developed using JXTA [7]. JXTA provides an open infrastructure allowing developers to create P2P networks operating across a variety of platforms. The original implementation of JXTA was developed in JAVA. JXTA-C and JXTA for mobile devices also exist. The default message distribution broadcasts messages to communicate with nodes. JXTA also supports a two tiered hierarchy for the overlay network, Which could be used to reduce the amount of traffic forwarded between nodes when discovering services.

2.3 Connecting JXTA with Services

In order for JXTA to interface with existing software running on hosts that provide network services, a mechanism was needed to hook into applications that need to communicate. This was accomplished through a generic wrapper around the process implementing the security service. Updates to the peer sent from an administrator or authorized node are processed by the peer. The peer reformats the control and configuration commands for the specific application executing the service. Commands are redirected to a configuration file and the application level process is restarted.

3 Implementation

Table 1 identifies several types of security services and how each maps to a particular platform.

Services were implemented for three types of network processing platforms: a Linksys Wireless Router, (model WRT54GS) [5], a workstation running Linux and a Global Velocity GVS1500 Extensible Network Switch [1]. The applications which run on these platforms include Intrusion Detection or Prevention, Quality of Service, and Anomaly Detection.

In this work we modified a Linksys wireless router to serve as a node in the network-wide managed infrastructure. The wireless router consists of a 54 MBit/sec WiFi connection, a 4 port 100Mbit/sec Ethernet switch and one external 100Mbit/sec uplink connection. The Linksys router uses a 200 MHz embedded processor with 32MBytes of RAM. An experiment has been discussed demonstrating the IDS software Snort [10] running on the Linksys router in [3]. This implementation of Snort on the Linksys router is limited in terms of the number of rules it supports and the rate at which it processes requests. The router provides native support for application and port based Quality of Service. Performing anomaly detection at the wireless router was not the intended purpose of the device and only works to a marginal degree because of limited CPU and memory resources. Anomaly services are better implemented using higher powered nodes.

A workstation configured with the Linux operating system is another platform we support for distributed management. Linux workstations are commonly used by network administrators implementing IDS. Networks running IDS software operate at rates measured in the 100's of Megabits/sec range. Two open source software tools perform IDS, Snort and Bro [9]. Linux systems can be used to provide some types of Quality of Service. The Hierarchical Token Bucket (HTB) packet scheduler [2] is found in standard Linux distributions from 2.4.20 and up. Anomaly detection can be performed using Linux with the Statistical Packet Anomaly Detection Engine (SPADE) [4]. SPADE is a Snort preprocessor plugin which sends alerts through the standard Snort reporting mechanisms.

The Global Velocity GVS1500 system uses the FPX hardware platform [**?**] to process packets in reconfigurable hardware at Gigabits/second rates. is a reconfigurable hardware device able to process traffic at gigabit link rates using FPGA's. The device also contains a single board computer executing software applications. FPGA Circuits have been developed for FPX modules that process Snort rules operating in both Intrusion Detection and Prevention modes. This work was presented in [6], and demonstrated Gigabits/sec header and payload processing of TCP streams. An application to detect worm activity was presented in [8]. This approach demonstrated the use of Bloom filters to maintain statistics on commonly occurring content. The GVS1500 also contains a single board computer that can be used to perform management functions.

4 Experimental Results

4.1 High level

In order to test the capabilities of the overlay network, experiments were performed to measure the overhead of JXTA and the publish/subscribe model for

	Wireless Router	Workstation	Extensible Switch
Intrusion Detection or Prevention	Snort with limited ruleset	Snort or Bro	FPGA Snort Lite
Quality of Service	Linksys QoS Support	HTB	FPGA Queue Manager
Anomaly or Event Detection	None	SPADE	FPGA Worm Detector

Table 1. Security Applications for each respective platform

nodes. Specifically, the time to process alerts and forward relevant information to nodes or groups of nodes interested in the alerts or aggregation of alerts. Figure 1 illustrates the network created to test the overlay network. The topology consists of an XML generator running on one host, a host which advertises and publishes alerts, the server, and clients interested in subscribing to the server. The XML generator creates XML alerts and sends them to the server. The server processes the XML alerts and forwards them to the subscribers.

4.2 Experiment Setup

The Emulab [11] environment is used to test the performance of the P2P overlay network. The nodes in the experiment consist of three 2Ghz Pentium 4 computers executing the Linux 2.4.20 kernel. The links between each node are set to 100Mbits/sec with 0ms delays between links. The XML generator injects 1000 144byte XML alert packets at various link rates. JXTA version 2.3.3 is used on all hosts. Communication between the nodes is through the JXTA API, using unreliable unicast pipes on top of TCP. JXTA pipes provide a mechanism to connect the input of one service (or node) to the output of another.

4.3 Results

Figure 2 represents the number of packets per second the generator is able to inject while avoiding packet loss at the client. The client is subscribed to one

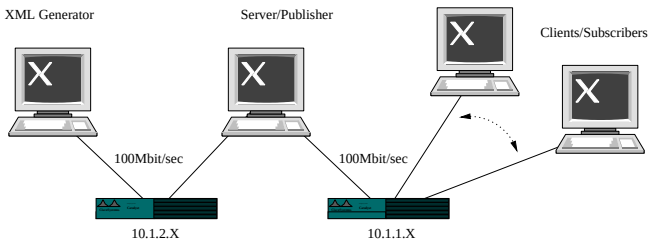

Fig. 1. Network topology constructed in Emulab

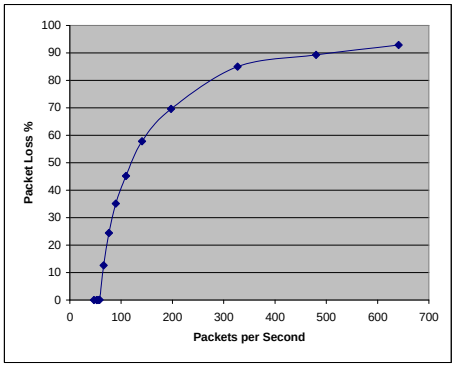

Fig. 2. Percentage of dropped as the number of packets per second increases

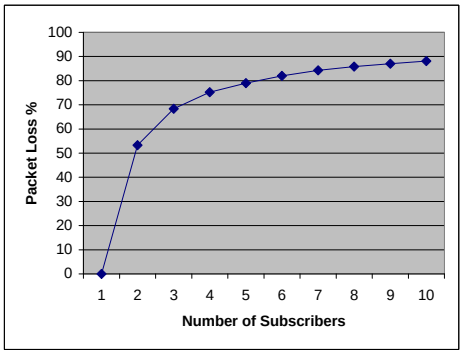

Fig. 3. Percentage of dropped packets as the number of subscribes increases

service, which consists of every alert from the XML generator. The number of dropped alerts increases with the increase of packets per seconds from the XML generator. At approximately 55 packets per second, JXTA is able to forward all 1000 alerts to a client.

Figure 3 illustrates the percentage of alert traffic dropped as the number of subscriptions increase. For each subscription an additional copy of the alert is generated. For example, when seven clients are subscribed, JXTA attempts to send out 1000 alerts to each of the seven clients (7000 total alerts). In this example, the clients are individual nodes, however they are capable of acting as rendezvous peers connected to a large group of nodes, similar to the top tier of a two tier overlay network. From the graph, we observe that with two subscriptions the number of alerts the server processes and forwards is cut almost in half. As the number of subscriptions increase, the total amount of alerts generated increases, however the number received at each client continues to decline.

During both of these experiments the CPU utilization of both the client and the server operated around 60-70%. The main reason for such low performance

stems from the amount of overhead necessary for a single JXTA alert message. In order to send a message, the server first creates a JXTA unidirectional output pipe to connect to the client. A reason for this is to ensure that the client is alive, as JXTA makes no assumptions about reliable nodes. The overhead associated with this however, is around 40ms per alert. The overhead of transmitting the JXTA alert was observed with tcpdump. Nine packets were transmitted between the client and server to create the output pipe and deliver the message.

Despite this drawback, optimizations exist to increase performance. The authors chose an implementation consist with examples provided by the JXTA tutorials. Maintaining individual state per client should increase overall performance. Keeping the connection open or queueing data at server are both optimization that should reduce the amount of overhead between the client and server.

5 Conclusion

The goal of this initial research is to investigate techniques for deploying services in the network for heterogeneous communications. Migrating to an open XML based solution imposes a fair amount of overhead as observed through the experiments. Characterizing the overhead and investigating appropriate uses of this technology is necessary. Moving forward, the goal is to developing more automated network management using open standards targeting network security.

References

1. Global Velocity. `http://www.globalvelocity.com/`
2. Heirarchical Token Bucket. `http://luxik.cdi.cz/devik/qpos/htb/`
3. SNORT on a WRT54G `http://www.batbox.org/wrt54g.html`, Sept. 2003.
4. Spade - Statistical Packet Anomaly Detection Engine. `http://www.-computersecurityonline.com/spade`, 2004.
5. Linksys. `http://www.linksys.com`, 2005.
6. M. Attig, S. Dharmapurikar, and J. Lockwood. Implementation results of bloom filters for string matchings. In *FCCM*, Napa, CA, Apr. 2004.
7. B. Traversat, et al. Project JXTA 2.0 Super-Peer Virtual Network.
8. B. Madhusudan and J. Lockwood. Design of a system for real-time worm detection. In *Hot Interconnects*, pages 77–83, Stanford, CA, Aug. 2004.
9. V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(23–24):2435–2463, 1999.
10. M. Roesch. SNORT - lightweight intrusion detection for networks. In *LISA '99: USENIX 13th Systems Administration Conference*, Seattle, Washington, Nov. 1999.
11. B. White, 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 *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002. USENIX Association.