NetSecu: A Collaborative Network Security Platform for in-network Security

Xinming Chen and Beipeng Mu
Department of Automation
Tsinghua University
Beijing, China
Email: {chen-xm09, mbp07}@mails.tsinghua.edu.cn

Zhen Chen
Research Institute of Information Technology (RIIT)
Tsinghua National Lab for Information Science and Technology
Beijing, China
Email: zhenchen@tsinghua.edu.cn

***Abstract*—Malicious attacks are frequently launched to make specified network service unavailable, compromising end hosts for political or business purpose. Though network security appliances are widely deployed to resist these attacks, there is a lack of dynamic and collaborative platform to flexibly configure and manage all the security elements. In this paper, we present NetSecu, a platform based on Java and Click Router, which can dynamically enable, disable and configure security elements such as firewall, IPS and AV. Furthermore, a collaborate module is implemented to integrate individual NetSecu platform into a Secure Overlay Network, providing collaborative traffic control against DDoS attack. Equipped with collaborate module, NetSecu platforms are organized in a tree hierarchy where each level node is registered to its father node. A Central Management Site acts as the root node for large scale deployment. The policy is distributed from higher level to lower level NetSecu nodes, while security events are aggregated from lower level to higher level. Performance evaluation shows that our NetSecu system can achieve line rate with and without security function. Finally we deploy the NetSecu platform in multiple sites, where our design is fully demonstrated and tested.**

***Keywords*-network security; collaborative security; click router; Java; overlay network;**

I. Introduction

Network security appliances, such as Firewall, IDS and most recently Unified Threat Management (UTM) [1], are widely deployed in vantage points and play an important role in protecting the network from attacks. Most of these appliances work without collaboration, their detection results are isolated, and cannot be collected and analyzed systematically. But for Internet nowadays, attacks occured on different sites may be closely related. With a global view of security events, the spread of virus can be easily detected, and DDoS attacks can be prevented more quickly and effectively.

In this paper we present NetSecu, a collaborative network security platform. It is a building block of a larger scale collaborative system. New security functions such as firewall, Intrusion Detection System (IPS) and antivirus (AV) can be deployed on NetSecu nodes at runtime, and can be dynamically enabled, disabled and upgraded. NetSecu is based on commodity hardware and commonly used Java with Click router. It can be easily commoditized. Compared with existing network security appliances, NetSecu consists of the following features:

1) Incrementally deployable security elements;
2) Dynamically enable / disable / upgrade security elements;
3) Policy-instructed collaboration over the Internet.

The rest of this paper is organized as follows: Section II introduces related works such as NetServ and POR. Section III presents the detailed design and implementation of NetSecu. Section IV evaluates the performance of a single NetSecu node. Section V presents the collaborative policy management of NetSecu. Section VI presents single-site and multiple-site deployment. As a summary, in Section VII, we state our conclusion.

II. Related Works

A related work is NetServ [2], a dynamically deployment in-network service. It combines the click router and Java, such combination meets the flexibility and stability we need in our project. Compared with NetServ, we pay more attention to network security features, and the performance issue incurred by Java.

Another similar work is Programmable Overlay Router (POR) [3] from Cisco. POR also use Java language to construct the in-network service and deploy Service Node (SN) in core network. From the perspective of a service construction, they also prefer a high-level language-based VM such as Java, because Java provides features such as isolation and easy deployment.

III. NetSecu Detailed Design

A. Overview

NetSecu is a collaborative network security device which can be used as a distributed attack detector and traffic controller. It is the building block of collaborative network security system which can visualize the security event and react to resist attacks in a consistent and unified way. Besides security functions, NetSecu platform can also be used as a traffic probe for forensics. Fig. 1 shows the role that NetSecu nodes play in an in-network environment.

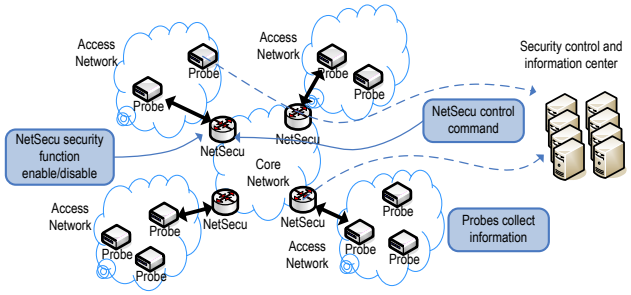

Figure 1. Collaborative in-network edge security platform.

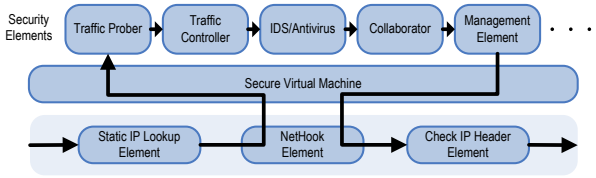

Figure 2. System software structure for each NetSecu node.

B. Framework Structure

A NetSecu node runs on Linux, and is based on user-space Click modular router and Java. The NetSecu software structure is shown in Fig. 2. The core of NetSecu is the NetHook element which captures the packet in-line and transfer to *Secure Virtual Machine* or *SVM* to process. Session table is organized as core data structure, and is maintained by SVM (built on Java VM). Security elements register themselves to SVM with event notification such as packet arrival. The security elements are isolated, so they can be dynamically enabled and disabled.

For example, a traffic measurement element is added to collect the traffic information and reports to security control center. With a global view of traffic characteristics in an ISP's networks, abnormal activities like DDoS are much easier to detect. Once detected, the filter module can be activated in NetSecu platform to control the DDoS traffic.

There have long been concerns about Java's performance, especially in computing-intensive applications such as IPS and AV. In order to inherit Java's dynamic deployment with high performance, Java Native Interface (JNI) is used to invoke C/C++ codes in the core applications. For example, our AV module uses ClamAV binary library for easy implementation and the speed of native code.

C. Security Elements

Security elements are the individual security functions for packet processing based on SVM. For better collaboration and control, we focus on the following elements besides other common network security elements,

1) Traffic Probe: A traffic probe is the building block for recording the raw Internet traffic in connection level. Hyperion [4], Time Machine [5] and NProbe [6] are all well known representative project in this function area.

Traffic probe is a Java element running on SVM. It can be designed to focus on special traffic incurred by security event.

2) Traffic Controller: A traffic controller controlls Internet traffic according to QoS requirements and application protocol identification. Traffic controller is also a Java element running on SVM, and can be dynamically enabled and configured according to security events.

3) Collaborator Element: A collaborator manages other security elements based on Security Center's command. It unites individual NetSecu platforms into a Secure Overlay Network. The communication command between NetSecu nodes and the security center is transmitted in a SSL channel to ensure security. A collaborator can start or stop a security element at runtime. Collaborators can respond to security event such as limiting the DDoS traffic on demand.

4) Reporting Element: Reporting element collects running log and threat events, and generates human-readable report. Based on the traffic logged by separate NetSecu nodes, traffic anomaly detection can be performed to discover large scale network anomaly.

Event aggregation is an important procedure after the events are reported from different NetSecu nodes. The NetSecu nodes are organized in a hieratical tree structure and the events will be reported from the low level to high level, which is shown in Fig. 8.

5) Local Manager and Interface: A local manager is responsible for local maintenance and management of security elements. The local manager also assigns the local administrator the right to configure and problem-diagnosis for system event.

D. Self Security of NetSecu Node

An individual NetSecu node consists of the followings system self defense approaches:

1) Bypass Function: An individual NetSecu node has bypass function to guarantee the normal usage in case of system fault. According to NetSecu node's software architecture in Fig. 2, there are three levels of bypass function: network interface hardware level (known as hardware cut-through), SVM level (direct routing forwarding in OS) and security element level (bypass the specified security function, e.g. AV, IPS etc.). The system status change is monitored by hardware or software watchdog. When there is a link failure or program fault, bypass function will be enabled, and the security functions are bypassed. Such function makes NetSecu more intelligent to resist unpredictable system fault or component fault.

2) Self DDoS Resist Function: A traffic control scheme based on IP active rate is implemented to make system more reliable to DDoS attack. This scheme is borrowed from Shield component of Untangle UTM [7]. The system

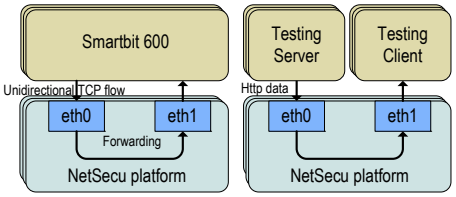

Figure 3. The test environments.

resource consumption is evaluated for each IP, including TCP connection numbers, memory size, flow numbers of each source IP, and give an evaluation value EV for each IP ($0 < EV < 100$) which reflects the degree of system cost and the probability of being a DDoS traffic. There are three actions for traffic control, for most low-cost traffics ($EV < 40$), NetSecu will forward them without any restrictions, otherwise, for middle level ($40 \leq EV \leq 80$), NetSecu will block some traffics or forward them with some limits, and if $EV > 80$, which means the traffic of some IP consume much more system resources than other IP does, NetSecu will drop all the traffics from this IP in order to ensure it would not cost too much to inhibit other traffics. This restriction also makes NetSecu itself resistant to potential DDoS attack. An interesting related work need to be mentioned is [8], which also take the future system resource consumption into account by proactively evaluating the system resource consumption for inspecting certain traffic, and temporally blocking or passing such traffic.

IV. Performance Evaluation for single NetSecu Node

A. Hardware Platform and Test Environments

NetSecu is deployed on different hardware platforms in order to compare its performance under different usage scenarios. The hardware configuration is shown in Table I.

We have two test methods here. A Smartbit 600 is used to test the forwarding rate of each hardware platform. A http server and a client are used to test the performance with security elements enabled. These two test environments are shown in Fig. 3.

B. MLFFR of NetSecu and Bare Linux

To ensure that NetSecu does not incur unacceptable performance decrease, the Maximum Loss-Free Forwarding Rate (MLFFR) of a NetSecu Node and bare Linux is evaluated. MLFFR is the highest forwarding rate with zero packet loss. As we will compare the security elements performance in Section IV-C, which is payload sensitive, we use Mbps here instead of packets/s in this test.

The bare Linux runs Debian 5.0 with kernel version 2.6.26. It is configured as a kernel router by setting the value of `/proc/sys/net/ipv4/ip_forward` to 1. NetSecu

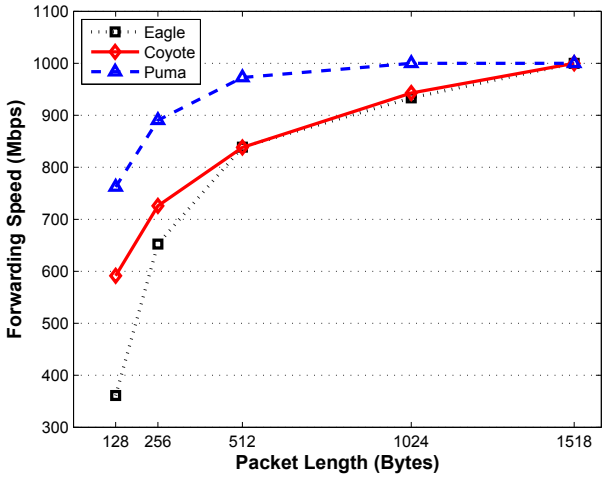

Figure 4. Forwarding performance of bare Linux (unidirectional).

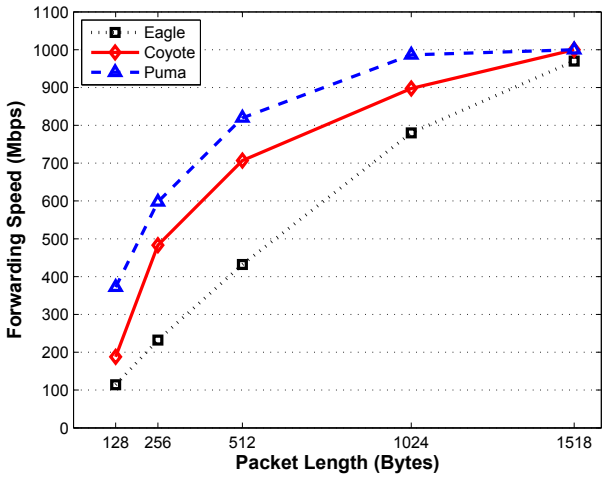

Figure 5. Forwarding performance of NetSecu (unidirectional).

is installed on the same OS. During this test, no security elements are enabled, as we only want to evaluate the framework overhead incurred by Java and Click.

The test is conducted as follows: a Smartbit 600 sends 100 unidirectional TCP flows to the input interface of a node, and receives the forwarded flow. Packet lengths varied from 128 bytes to 1518 bytes. All the three types of hardware are tested, the result is shown in Fig. 4 and Fig. 5.

From the result it can be seen that the overhead when comparing bare Linux and NetSecu is sizable for short packets, but for packets longer than 512B the forwarding performance is not significant. This is expected because Java VM has limited execution efficiency, and the kernel-to-user packet transition is another significant overhead. Such overhead is acceptable, because according to our statistic, the

Table I
HARDWARE PLATFORMS TESTED WITH NETSECU

Code Name	CPU	Memory	Chipset	NIC
Eagle	Intel Core 2 T7200, 2.0GHz	4GB DDR2, 800MHz	Intel 945 + ICH7R	Intel 82573L
Coyote	Intel Core 2 Q9400, 2.66GHz	4GB DDR3, 1066MHz	Intel G41 + ICH7R	Intel 82574L
Puma	2×Intel Xeon E5504, 2.00GHz	8GB DDR3, 1066MHz	Intel 5520 + ICH10R	Intel 80003ES2LAN

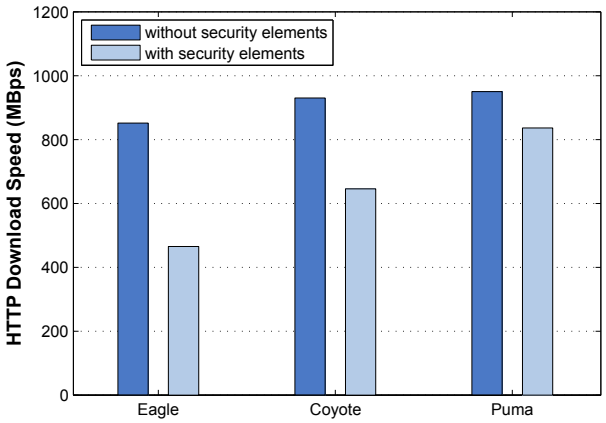

Figure 6. Processing speed comparison with and without security function.

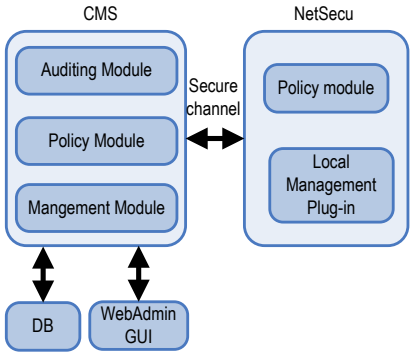

Figure 7. Central management and collaborative policy generation and distribution.

average packet size of normal traffic is around 512B, so that the performance of NetSecu under real network environment can be guaranteed.

C. Performance of Security Elements

In order to test the performance of security elements such as IPS and AV, HTTP traffic is used in this test. We configure two servers as a HTTP client and a HTTP server. The client starts 20 threads, continuously request and download a 20MB file from the server. The security elements are firstly disabled and then enabled. The result is shown in Fig. 6.

The http download speed with security elements enabled is significantly lower than the speed with security elements disabled, especially for Eagle platform. This is expected because security functions needs a lot of computation power. Puma platform suffers less performance decrease, because its forwarding rate is limited by its interface speed, the actual forwarding ability is beyond 1Gbps. When the Puma platform executes computationally intensive tasks, the bottleneck turns to be CPU, so there is a slightly performance decrease with security function, which is acceptable. This experiment shows that NetSecu can provide enough throughput with enhanced hardware.

In NetSecu prototype, as performance is not the most urgent consideration, we are not trying to improve NetSecu's throughput and latency besides some common tricks in Linux network stack [9]. We believe there is still much room for performance optimization.

V. COLLABORATIVE POLICY MANAGEMENT

A. CMS model

While NetSecu can be deployed in single site as a security gateway, it is also capable for multiple-site deployment. In order to better organize the distributed NetSecu nodes, a *Central Management Site (CMS)* is added into the collaboration system. It is responsible for collaborative policy distribution, event aggregation and NetSecu status monitoring. The CMS model is shown in Fig. 7.

The CMS implementation has three modules. Firstly, auditing module is provided for auditing security events which are reported and collected by each local NetSecu node; all the reports are saved in central database. Secondly, Policy module is designed to set security policy for security elements through each NetSecu node, administrators can use GUI to modify each policy. Thirdly, management module plays a role of a central controller. It provides uniform management of each local NetSecu node, which can monitor running state of each security element and make dynamic security configuration.

For the purpose of large scale deployment, NetSecu nodes are organized in a tree hierarchy shown in Fig. 8. Lower level NetSecu nodes (leaf nodes) are registered in higher level NetSecu nodes, and the CMS acts as root node. The policy is applied downward from higher level to lower level NetSecu nodes, while the events are aggregated from lower level to higher level.

B. NetSecu with Nagios

Nagios [10] is a widely deployed network monitoring system that enables administrators to identify and resolve

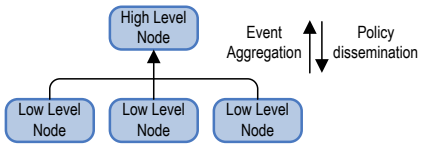

Figure 8. NetSecu tree hierarchy.

computing cluster problems in mission-critical infrastructure processes. Nagios provides users with instant status of remote servers. It can be used to detect and repair problems and mitigate fault issues.

Because of the scalability and flexibility of Nagios, it is used in our collaborative policy management implementation. We also enhance Nagios with our specific modification in the exchange message format and private commands in our cluster management of NetSecu nodes.

C. Message types and commands

A NetSecu node communicates with CMS in the following Nagios format:

```
define command{
  command_name cmd
  command_line $USER1$/ cmd
    -H $NetSecure_ADDRESS$
    -c $ARG1$ -a $ARG2$
}
```

The NetSecu node parses the message, and interprets the command and their parameters, and executes the command in console accordingly:

```
command[cmd]= CMD  $ARG1$
  -c $ARG1$ -a $ARG2$
```

The commands are categorized into groups of secure functions, system configuration, logging, policies management and status reporting. The detailed command list is shown in Table II.

VI. Deployment

A. Single Site Deployment

Used as a security gateway, NetSecu is tested in Capital Info Network, an ISP running Beijing Capital-Info E-Government networks. There is an amount of 700 to 800 servers in this network, including web servers, database servers, etc. The entrance bandwidth is 1Gbps and the average traffic (mostly http traffic) is about 500Mbps. The deployment topology is shown in Fig. 9.

With the feature of dynamically start and configure elements, NetSecu has been running for six months without interrupting normal traffic.

B. Multiple Site Deployment

A test bed is constructed to validate the effect of multi-site deployment of NetSecu platform. There are four sites includes Beijing Capital-Info network, IDC Century-Link

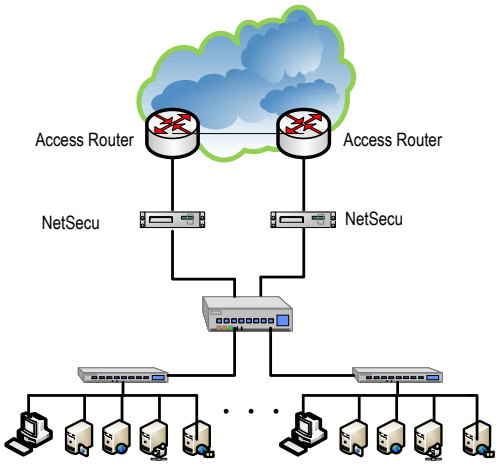

Figure 9. Single site deployment with HA backup.

network, an enterprise network and a campus network. The four sites are connected over the Internet. A reference deployment is shown in Fig. 10.

In each site, there are several NetSecu nodes which take charge in different network environment to adapt to different link speed, e.g. NetSecu with Eagle and Coyote platforms handles fast Ethernet and Giga Ethernet traffic respectively. Also there is a CMS that connects to each Netsecu node.

During the two months' running, the collaborative mechanism runs as we expected. Rule sets are dispatched correctly. A large volume of reports from each site have been collected. There are a lot of network security problems have been observed and recorded in our test bed, such as DDOS reflect attacks, Spam scatter and ad hoc P2P protocols etc.

VII. Conclusion

In this paper, we focus on how to effectively manage various security elements in production network. Based on this demand, we design and implement NetSecu, a network security management platform which can dynamically enable and configure security elements. Each NetSecu node is built based on Java and Click Modular Router. By exploiting this combination, we can dynamically enable, disable and upgrade security elements at run-time. Performance evaluation shows that NetSecu can provide enough performance for commonly-used network security management platform.

For large scale deployment, our NetSecu is organized as a tree hierarchy, where configuration policy can be set downward to lower level nodes, and events can be reported upward to higher level nodes. NetSecu is designed to integrate different NetSecu nodes from all the sub-networks and form a uniform and scalable in-network security system.

Finally we deploy and operate the NetSecu platforms in a production network and construct multiple site environments for test purpose, and demonstrate the feasibility of our design and deployment.

Table II
COLLABORATIVE MANAGEMENT COMMANDS UTILITY

Secure Element	System Configuration	Reporting	Logging	Policy management	Misc
CLI active function-name	CLI active	CLI enableReporting	CLI startLog	CLI addPolicy	CLI shutdown
CLI deactive function-name	CLI deactive	CLI prepareReports[args]	CLI stopLog	CLI delPolicy	CLI platformStats
CLI list function-name[args]	CLI upgradable	CLI startReports	CLI restartLog	CLI listPolicy	CLI who
CLI start function-name	CLI uptodate	CLI stopReports	CLI clearLog	CLI updatePolicy	CLI getAdmin
CLI stop function-name					CLI passwd[-add|-del]
CLI kill function-name					
CLI update					
CLI upgrade					

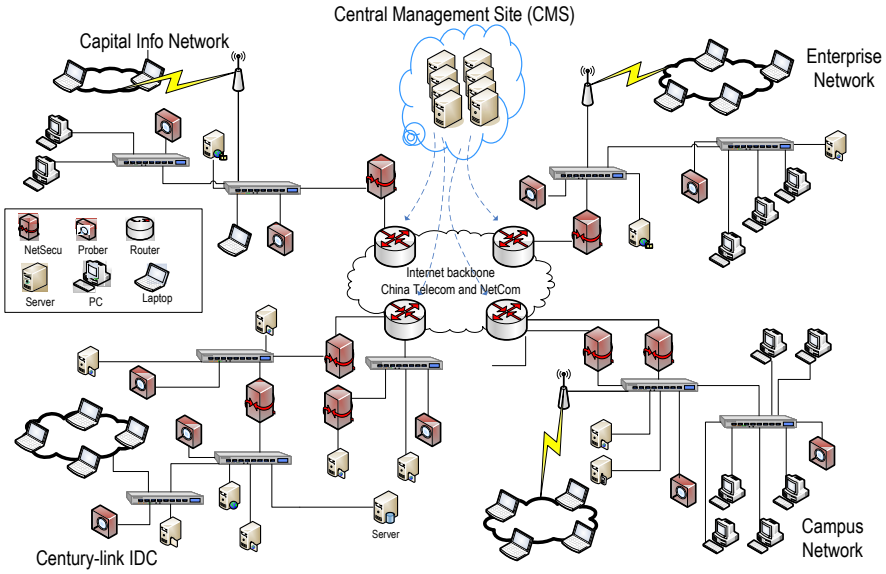

Figure 10. Multiple site deployment.

ACKNOWLEDGMENT

This project is supported by National High-tech Program No. 2007AA01Z468 with the title of *A Holistic UTM System Design and Implementation*.

This project is also supported by Beijing Municipal Science and Technology Division and HOSUN Tech.

Special thanks to QoS Lab of CS department for providing Intel Nehalem platform and their generous help.

REFERENCES

[1] "Unified threat management appliances and identity-based security: The next level in network security," *IDC Go-to Market Services*, September 2007.

[2] S. Srinivasan, J. W. Lee, E. Liu, M. Kester, H. Schulzrinne, V. Hilt, S. Seetharaman, and A. Khan, "Netserv: Dynamically deploying in-network services," in *the 5th International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2009.

[3] B. Davie and J. Medved, "A programmable overlay router for service provider innovation," in *Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO)*, Aug. 2009.

[4] P. J. Desnoyers and P. Shenoy, "Hyperion: High volume stream archival for retrospective querying," in *USNIEX*, 2007.

[5] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider, "Enriching network security analysis with time travel," in *the ACM SIGCOMM*, 2008.

[6] L. Deri, "Modern packet capture and analysis: Multi-core, multi-gigabit, and beyond," in *the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2009.

[7] "Untangle," *HTTP://www.untangle.com*.

[8] P. Barlet-Ros, G. Iannaccone, J. Sanjuàs-Cuxart, D. Amores-López, and J. Solé-Pareta, "Load shedding in network monitoring applications," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, 2007, pp. 5:1–5:14.

[9] S. Tripathi, N. Droux, T. Srinivasan, K. Belgaied, and V. Iyer, "Crossbow: A vertically integrated qos stack," in *Workshop on Research on Enterprise Networking (WREN)*, 2009.

[10] "Nagios," *HTTP://www.nagios.org*.