Jobber: Automating Inter-Tenant Trust in The Cloud

Andy Sayler Eric Keller Dirk Grunwald
University of Colorado, Boulder

Abstract

Today, a growing number of users are opting to move their systems and services from self-hosted data centers to cloud-hosted IaaS offerings. These users wish to both benefit from the efficiencies that shared multi-tenant hosting can offer while still retaining or improving the kinds of security and control afforded by self-hosted solutions. In this paper, we present Jobber: a highly autonomous multi-tenant network security framework designed to handle both the dynamic nature of cloud data centers and the desire for optimized inter-tenant communication. Our Jobber prototype leverages principals from Software Defined Networking and Introduction Based Routing to build an inter-tenant network policy solution capable of automatically allowing optimized communication between trusted tenants while also blocking or rerouting traffic from untrusted tenants. Jobber is capable of automatically responding to the frequent changes in virtualized data center topologies and, unlike traditional security solutions, requires minimal manual configuration, cutting down on configuration errors.

1 Introduction

We are in the midst of a paradigm shift in the way we host our digital infrastructure. The rise of cloud computing has driven users away from private, self-hosted approaches toward large, multi-tenant data centers that are both more cost-efficient and more flexible than traditional solutions. Security in these public data centers is a significant concern, and many researchers have focused on designing technology for isolating tenants. This tendency toward strict isolation, however, leaves untapped benefits – namely, numerous possibilities for inter-tenant optimization and cooperation derived from the collocation of interacting services in multi-tenant data centers. Inter-tenant communication in modern data centers already composes 10% to 40% of all data-center communication [7], and this number will likely grow as we continue to consolidate data centers and shift services into the cloud. It is thus desirable to find more efficient ways to process and secure such traffic, leading to both an increase in cloud performance and a decrease in operation cost.

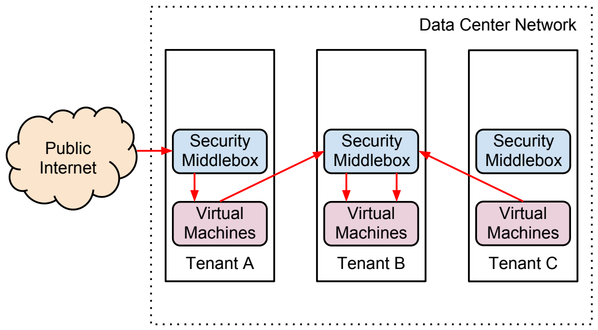

Figure 1: Inter-Tenant Data Communication Today

Consider a website that uses an advertisement bidding and insertion system. Even though the ad system likely shares tenancy with many of its clients in the data center, the service must treat all client interactions as untrusted communication, increasing network latency due to the overhead required to secure such communication. In the absence of manual configuration based on prearranged business-to-business relationships [11], inter-tenant communication is treated in the same manner as external communication: as untrusted traffic that must be passed through a complex and resource intensive security stack. This security stack often includes systems like inter-domain routers, firewalls [5], traffic monitors, and a variety of other middlebox systems (Figure 1). In the cloud, these systems require additional resources to run, costing tenants extra money and increasing network overhead. In addition to the costs and overhead incurred when treating all communications as untrusted traffic, traditional security solutions rely heavily on statically configured polices that are prone to misconfiguration. The highly dynamic nature of multi-tenant data centers only exasperates this issue, requiring frequent manual policy updates to handle the volatile topologies associated with virtualized infrastructure.

We are interested in removing this manual configuration and automating the development of trusted relationships. Trust relationships should not require manual configuration, but should instead grow organically through interaction and reputation – e.g., between the ad system and a well-behaving client. Toward this end, we present Jobber[1]: a highly dynamic network security system designed to handle both the volatile nature of the cloud and the desire for optimized inter-tenant communication in multi-tenant data centers.

Much like Ethane [9], Jobber can transparently direct untrusted traffic through high-overhead, high-security flows while automatically allowing trusted traffic to pass through optimized lower overhead flows. In contrast to Ethane, which relies on statically configured rules, Jobber builds and leverages a *trust network* between tenants to dynamically determine if a communication attempt between tenants should be allowed, denied, or routed through additional security infrastructure.

The Jobber architecture includes three core components. The individual implementations of each component are flexible (although we provide suggestions based on the development of our prototype); it is the relationships between the components and the core functionality provided by each that comprise our core Jobber vision:

Trust Network: Jobber employs a trust network between tenants to compute which tenants should be granted low-overhead, privileged connections, and which should be treated as untrusted hosts or blocked completely. Our Jobber prototype uses techniques from Introduction Based Routing (IBR) [10] to realize this capability, effectively placing a market value on good behavior. This encourages well behaved tenants to form inter-tenant trust relationships while isolating misbehaving tenants.

Sensor Framework: Jobber requires one or more sensor systems from which it can gather data on which to base individual tenant reputations within the trust network. Our prototype employs a modular sensor framework that we intend to interface with a variety of network data sources, from intrusion detection systems (IDS) to host-based access logs.

Programmable Routing and Security: Jobber must be able to programmatically control network routing and security rules in order to steer traffic between various network security paths (or block it all together). We have explored prototypes that utilize various programmatic routing and security schemes. Using legacy cloud infrastructures, Jobber can interact with various cloud data center APIs to control routing and security. In future SDN-based data centers, Jobber can interact directly with an OpenFlow-based SDN controller [13] to obtain this functionality.

[1]A jobber is a middle man in the exchange of stocks and securities.

In the remainder of this paper we discuss using reputation-based trust networks to obtain a largely configuration-less security framework (§2). We then explore the architecture of Jobber in §3, discuss open problems related to Jobber in §4, and conclude in §5.

2 Configuration-less Security

Most network security solutions in use today are statically configured: the user programs them with a predefined set of rules regarding who can and who cannot communicate. These rules must be constantly updated as the network topology and the nature of inter-tenant relationships change. Jobber moves away from this static configuration model in favor of an autonomous configuration model that can dynamically adapt to a changing network environment. In this section we discuss the problems with static configuration as well as the technologies Jobber employs to achieve autonomous operation.

2.1 Problems with Static Configurations

It is a well documented fact that manually configured network security systems are highly prone to configuration errors [16, 9]. These configuration errors are often the root cause of security breaches in "secure" networks. Many of these configuration errors stem from the fact that generating a policy stipulating all allowed (or denied) communication on even very simple networks can be a very complicated task. It is easy to forget to add a rule, or to add a rule too many. Even advanced modern firewall systems like Firmato [8] generally start with elaborate policy definitions.

And if properly elucidating the security policy on a static network isn't hard enough, the volatile nature of modern networks only serves to make the situation much, much worse. Today's networks, be they locally hosted or in a shared data center, are often populated by more virtual machines than physical ones. This virtual infrastructure gives operators a lot of flexibility in terms of reassigning computing resources in response to various demands, but this flexibility also means that network security polices must be constantly kept up-to-date with the current state of the network. Properly capturing the network's static security policy is no longer enough. We must now update these polices every time we create or destroy virtual resources, making already common configuration errors even more likely.

Furthermore, trust relationships between tenants change on a regular basis. A trusted partner one day may be a compromised and threatening partner the next. Systems that rely on manually configured trust networks based on semi-stable business relationships are unable to quickly respond to the shifting threat landscape inherent in modern networks and data centers.

2.2 Trust Networks and IBR

The solution is to move away from systems that rely on statically captured network policies. Instead, we want an organically grown, autonomous, and highly dynamic mechanism for capturing the security state of the network at any given instance. Trust networks seem a promising candidate to fit this bill. Researchers have proposed a variety of methods for dynamically building trust networks, from concepts that leverage social relationships [14] to concepts based on the behavior of power grids [12]. Introduction Based Routing (IBR) [10] is another novel approach to building dynamic trust networks. IBR relies on game-theoretic models of trusted communities. Communities dynamically grow and shrink on the basis of introductions between hosts. The system incentives making introductions to well-behaved hosts while discouraging the making of introductions to misbehaving hosts.

In IBR, a host must ask to be introduced to any host with which it wishes to communicate. Whether or not a host is willing to make an introduction is based on reputations that each host maintains for the other hosts with which it has interacted. When communication between two introduced hosts completes or is terminated, the involved parties send either positive or negative feedback to the host who originally introduced them, allowing that host to continue to build its internal reputation map. This system places a market value on good behavior, and eventually leads to the isolation of misbehaving hosts and the interconnection of behaving hosts.

The cost-structure of cloud data centers serves to further reinforce IBR's "market value on good behavior", making it an especially good candidate to satisfy Jobber's trust network component. Jobber extends IBR beyond per-host reputation to maintain reputation on a per-tenant basis. This system allows each tenant to maintain a reputation for each other tenant and to use these reputation as the basis for making introductions to allow direct tenant-to-tenant communication.

A system based on reputation raises a bootstrapping problem since it must start with no knowledge of prior interactions. To cope with this, we bootstrap the Jobber IBR network by making the data center provider a special case that will always provide an initial introduction between tenants. In effect, each tenant starts out with a minimally-trusted connection to the data center provider. They can then request introductions to any other tenant via the common provider node, noting that such connections have not been vetted in the same manner as introductions from other hosts, and should thus be treated as "probationary" until more data is available on which to gauge their reputation. Over time, communities of trust are formed as tenants develop more inter-tenant trust connections. As this happens, the reliance on the cloud provider as an introducer diminishes.

2.3 Sensor Framework

In order to determine reputations, sensors are needed to gather evidence as to whether or not a tenant is well behaved. We propose an extensible per-tenant Jobber sensor framework with a standardized interface that allows Jobber to collect information from a variety of sources and use it as the basis for IBR feedback generation (Figure 2). Possible sources of interaction information range from network-wide solutions like centralized intrusion detection systems [15], to host-based solutions like firewalls and authentication logs, to Jobber-aware application solutions that can directly report to the framework. For example, a tenant could configure the existing security logging system on a deployed host to report all connection attempts on closed ports to the Jobber sensor framework. The framework could then use this data to decrease the reputation of hosts attempting to make unauthorized connections. With this system, one must initially configure of the sensors and definitions of what constitutes "good" and "bad" behavior. Once initially configured, however, the framework automatically incorporates the sensor feedback to maintain the underlying reputation maps.

By centralizing IBR data collection for each tenant and exposing a standardized data collection interface, we can both aggregate feedback from a multitude of hosts as well as collect feedback from a variety of systems. This flexible approach fits well with Jobber's underlying goal of making data center security more flexible and autonomous. Our sensor approach is similar to that taken by centralized monitoring projects like Nagios [3] that rely on a standardized, extensible interface to collect data from a wide variety of information sources. After gathering all the data in a centralized location, each tenant can maintain their own generic lists of which data points constitute positive behavior and which constitute negative behavior. These data points can then lead to the single interaction feedback score provided to other tenants at the end of a given multi-tenant interaction.

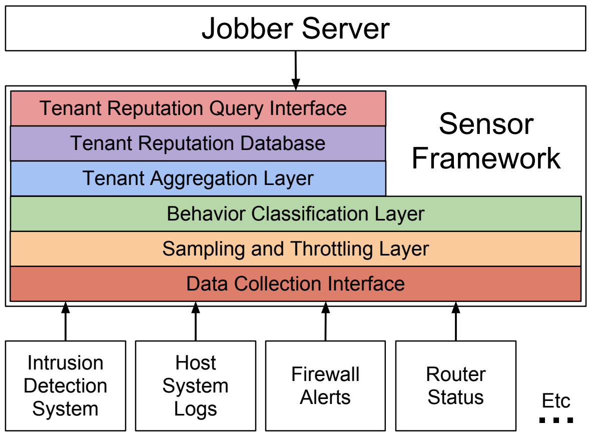
Jobber Server
Tenant Reputation Query Interface
Tenant Reputation Database
Tenant Aggregation Layer
Sensor Framework
Behavior Classification Layer
Sampling and Throttling Layer
Data Collection Interface
Intrusion Detection System
Host System Logs
Firewall Alerts
Router Status
Etc ...

Figure 2: Extensible Sensor Framework

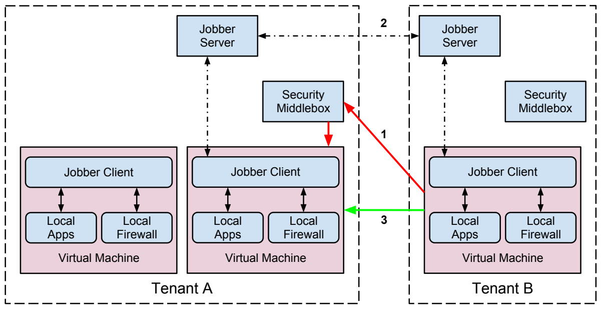

Figure 3: Jobber in a legacy data center with Jobber aware clients - *Flow 1 (red) shows a typical untrusted interaction being processed via the traditional security stack. Flow 2 (black) shows the Jobber servers for each tenant interacting to request a trusted connection. Flow 3 (green) shows a direct, trusted tenant-to-tenant interaction that bypasses the traditional security stack.*

3 Jobber Architecture

Jobber is an overarching system that orchestrates different resources to realize reputation based trust systems in multi-tenant cloud infrastructures. In this section we discuss three increasingly transparent and optimized Jobber architectures. In particular, we focus on architectures using the interfaces provided by Amazon's Elastic Compute Cloud (EC2) [4] service. We discuss both host-aware and host-unaware architectures. Then we discuss a Jobber architecture that leverages the OpenFlow framework to overcome the limitations inherent in the legacy architectures.

3.1 Today's Cloud - Host Aware

Our first architecture assumes that each host can be made Jobber aware. This requires running an additional Jobber process on each host, as well as interfacing specific applications with Jobber to provide feedback.

Figure 3 shows the basics of a client aware Jobber architecture. In this architecture, each tenant operates a Jobber server. This server is responsible for negotiating and responding to all Jobber related requests from other tenants. It also interacts with the Jobber client processes running on each of the tenant's hosts.

When a Jobber-aware application on Tenant B wishes to open an optimized, direct connection with a host from Tenant A, it must notify the Jobber client process of this request. The Jobber client then repeats the request to the Jobber server on Tenant B, who then contacts Jobber servers from other tenants to secure the necessary introductions to Tenant A. If a trust path can be successfully built, the Jobber server on Tenant A notifies the Jobber client on the host in question to open up the necessary host-based firewall ports to allow direct communication from the Tenant B host. The Tenant B Jobber Server then informs the application that originally requested the direct connection that the connection is now available, allowing the application to switch to using the direct connection, and avoiding the overhead normally imposed by the Tenant A security stack for untrusted communication.

This architecture allows tenants to leverage Jobber in data centers that do not yet offer support for programmatic network routing and security changes. Its reliance, however, on each individual host and application being Jobber aware still imposes a large overhead to deployment. We can avoid this overhead if we utilize an architecture that does not require client awareness.

3.2 Today's Cloud - Host Unaware

Figure 4 shows an alternate Jobber architecture for existing data centers that does not require each individual host and application to be Jobber aware. This architecture leverages Amazon's Virtual Private Cloud (VPC) [6] service to dynamically reroute host communication along either trusted or untrusted connections. VPC provides functionality for programmatically updating the routing table for a tenant's network, allowing the Jobber server to manipulate the routes as required to obtain the desired functionality. This architecture also introduces the Jobber sensor interface discussed in §2. This interface is used to both monitor trusted interactions for the purpose of generating feedback as well as to monitor untrusted connections to identify connections that might be benefit from promotion to trusted status.

In this architecture, the Jobber server, using information from the Jobber sensor interface, decides when it would be beneficial to promote an inter-tenant interaction to trusted status. When Tenant B's Jobber server finds a potential trusted interaction, it communicates with Job-

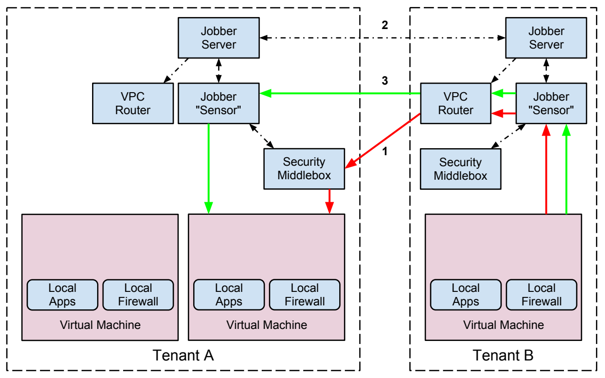

Figure 4: Jobber in a legacy data center without client modification - *Labeled flows are as they are in Figure 3 with the addition of the Jobber sensor interface monitoring traffic and the programmable VPC router directing traffic between trusted and untrusted paths.*

ber servers for other tenants to obtain the necessary introductions and establish the required trust connection to Tenant A. If a direct trusted connection can be obtained, Tenant A opens up the necessary firewall ports to allow a direct connection from Tenant B. It also sets up the necessary sensor infrastructure to monitor the trusted connection for the purpose of providing post-interaction feedback. It then informs Tenant B that the direct connection is available. The Tenant B Jobber server updates its VPC routing table to begin steering the newly promoted inter-tenant interaction directly to the Tenant A host, bypassing the standard Tenant A security stack.

These EC2-based architectures, while appropriate for deployment in an existing cloud infrastructure, still have a number of downsides. In particular, the bootstrapping methods discussed on §2 must be modified in order to compensate for the lack of cloud provider involvement. Furthermore, these architectures either require Jobber-aware hosts and applications or require additional infrastructure to handle the detection of optimize-able inter-tenant flows and to direct inter-tenant routing.

3.3 Jobber for the Future Cloud

We can overcome the limitations of existing cloud infrastructures by architecting Jobber for future data centers. We imagine these data centers offering Jobber awareness as a service, optimized for their SDN-capable back-end networks. We can leverage the SDN capabilities of the future data center to transparently detect and direct traffic at a data-center wide level, avoiding the need for per-tenant Jobber routers and proxies. This future data center gives us the ability to realize a Jobber architecture that is both more capable and less expensive than the legacy architectures discussed thus far.

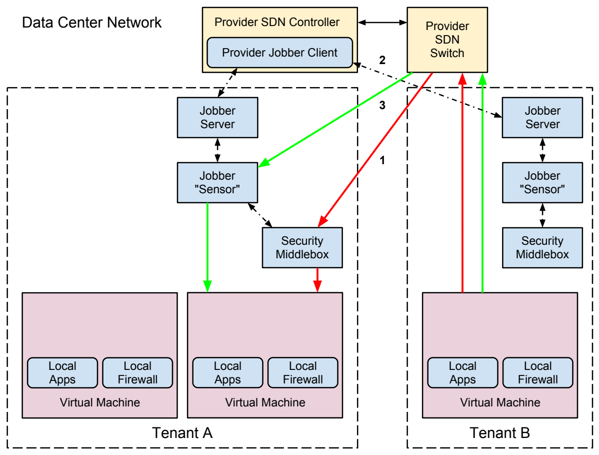

Figure 5: Jobber in an SDN-capable, Jobber-aware data center - *The flows are as described in Figure 4, except that the VPC router has been replaced by a Jobber-aware data center switch/router, and that the data center controller is now Jobber aware and capable of transparently creating direct connections whenever possible*

Figure 5 shows our Jobber architecture integrated with an SDN-capable data center. As in the legacy architectures, each tenant operates a Jobber server responsible for responding to introduction requests. This server interacts with the Jobber sensor framework to collect data related to ongoing inter-tenant communication and tenant reputation. Unlike previous architectures, however, we now exploit the SDN capabilities of the data center network by running the Jobber client as an application on the data center SDN controller. In this way, the data center controller can respond to “Packet In” events caused by attempted inter-tenant flows by attempting to secure a direct trusted connection between communicating tenants.

For example, when a host from Tenant B attempts to communicate with a host from Tenant A, the data center controller, via the Jobber client, queries the necessary per-tenant Jobber servers for the required introductions between Tenant B and Tenant A. If a trusted introduction sequence is available, the controller writes a flow to the data center switch forwarding the communication directly between tenant hosts and bypassing the untrusted security stack for each tenant. If no trusted connection is available, the switch forwards the communication along the untrusted path where Tenant A can either choose to block it all together or to allow it subject to additional security measures.

This arrangement provides a low overhead means for deploying Jobber. Each tenant maintains the flexibility to monitor reputation metrics and allow or deny introduction requests as they see fit, while the controller shoulders the burden of finding the necessary introduction sequences and routing the traffic along the appropriate path. In this deployment, we envision data center providers offering the Jobber framework as a service which tenants can leverage by simply standing up a Jobber server to handle all incoming Jobber requests. If a tenant takes advantage of this service, they can save money and effort by enabling dynamic, trusted, low-overhead communication with other trusted tenants.

4 Open Problems and Questions

Jobber is currently a work in progress. We have built a Jobber proof-of-concept prototype for the SDN-based architecture proposed in §3 using the Floodlight SDN controller framework [1] and the Mininet network emulation platform [2]. We are working on expanding our SDN-aware prototype toward the full functionality discussed in this paper, as well as constructing prototypes of the alternative legacy architectures proposed using Amazon EC2. As a work in progress, there are still a number of open problems and questions related to our Jobber vision.

The biggest open question is exactly how much of a performance and cost benefit Jobber can provide. We know that inter-tenant traffic comprises a significant fraction of total data center traffic, but until we can deploy a fully functioning Jobber prototype and evaluate its performance relative to the existing solutions, it is difficult to qualify exactly how much of a performance gain Jobber has to offer. While we are confident that Jobber will provide both performance and cost benefits, we also note that Jobber offers automatic configuration benefits regardless of any performance and cost advantage.

We must also complete a more thorough analysis and definition of the Jobber security model. Trust networks like IBR are effective in determining who is trustworthy and who is not, but they do tend to allow for one "free attack" per tenant when a previously trustworthy tenant begins to misbehave. How quickly Jobber can detect and respond to these trusted-to-untrusted reputation flips and how much damage might be done before it does remains to be seen.

Finally, while we propose candidates to satisfy each of the three core Jobber components discussed in §1, it is possible that there are alternate implementations for each component that may work better than our currently proposed suggestions. Once we complete our initial Jobber prototype using the discussed implementation (IBR, SDN, client-defined sensor framework), we plan to complete several alternate implementations and evaluate their relative merits. We also believe that it may be possible to push down some of these component implementation decisions to individual tenants (as we have tried to do with our sensor framework) in order to keep Jobber as flexible as possible and to encourage its use in a variety of diverse situations.

5 Conclusions

We believe that Jobber provides a compelling alternative to the existing configuration intensive static security paradigm used in many cloud data centers. Jobber allows for optimized inter-tenant cooperation and communication not subject to the security overhead required for untrusted connections. Jobber's ability to leverage optimized inter-tenant communication helps reduce the cost of using shared infrastructure while securely increasing the throughput available in such infrastructure. Jobber's freedom from manual configuration makes shared infrastructure more manageable while decreasing the security vulnerabilities introduced by configuration errors.

References

[1] Floodlight. `http://floodlight.openflowhub.org/`.

[2] Mininet. `http://mininet.github.com/`.

[3] Nagios. `http://www.nagios.org/`.

[4] AMAZON. Amazon ec2. `http://aws.amazon.com/ec2/`.

[5] AMAZON. Amazon ec2 security groups. `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html`.

[6] AMAZON. Amazon virtual private cloud. `http://aws.amazon.com/vpc/`.

[7] BALLANI, H., JANG, K., AND KARAGIANNIS, T. Chatty Tenants and the Cloud Network Sharing Problem. *Proc. of NSDI* (2013).

[8] BARTAL, Y. Firmato: A novel firewall management toolkit the hebrew university of jerusalem. *Proceedings of the 1999 IEEE Symposium on Security and Privacy 22*, 4 (2004), 381–420.

[9] CASADO, M., FREEDMAN, M., AND PETTIT, J. Ethane: Taking control of the enterprise. *ACM SIGCOMM Computer Communication Review 37*, 4 (2007).

[10] FRAZIER, G., DUONG, Q., WELLMAN, M., AND PETERSEN, E. Incentivizing responsible networking via introduction-based routing. *Trust and Trustworthy Computing 6740* (2011).

[11] IBM. Vpn scenario: Basic business to business connection. `http://tinyurl.com/ibm-vpn-b2b`. Access March 4th, 2013.

[12] MAO, Y., SHEN, H., AND SUN, C. From credit and risk to trust: towards a credit flow based trust model for social networks. *Proceedings of the 17th ACM international conference on Supporting group work* (2012), 209–218.

[13] MCKEOWN, N., AND ANDERSON, T. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review 38*, 2 (2008).

[14] MISLOVE, A., POST, A., DRUSCHEL, P., AND GUMMADI, K. Ostra: Leveraging trust to thwart unwanted communication. *Proc. of NSDI* (2008).

[15] PAXSON, V. Bro: a system for detecting network intruders in real-time. *Computer networks 31*, 23-24 (1999).

[16] WOOL, A. A quantitative study of firewall configuration errors. *Computer 37* (2004), 62–67.