Understanding Security Group Usage in a Public IaaS Cloud

Cheng Jin*, Abhinav Srivastava†, Zhi-Li Zhang*
*Computer Science Dept., University of Minnesota, {cheng,zhzhang}@cs.umn.edu
†AT&T Research, {abhinav}@research.att.com

***Abstract*—To ensure security, cloud service providers employ security groups as a key tool for cloud tenants to protect their virtual machines (VMs) from attacks. However, security groups can be complex and often hard to configure, which may result in security vulnerabilities that impact the entire cloud platform. The goal of this paper is to investigate and understand how cloud tenants configure security groups and to assist them in designing better security groups. We first conduct a measurement-based analysis of security group configuration and usage by tenants in a public IaaS cloud. We then propose and develop a tool called *Socrates*, which enables tenants to visualize and hence understand the static and dynamic access relations among VMs. *Socrates* also helps diagnose potential misconfigurations and provides suggestions to refine security group configurations based on observed traffic traversing tenants' VMs. Applying *Socrates* to all tenants hosted on the public IaaS cloud, we analyze the common usage ("good" as well as "bad" practices) of cloud security groups and report the key lessons learned in our study. To the best of our knowledge, our work is the first to analyze cloud security group usage based on real-world datasets, and to develop a system to help cloud tenants understand, diagnose and better refine their security group configurations.**

I. Introduction

In Infrastructure-as-a-Service (IaaS) cloud platforms such as Amazon EC2 and Openstack [1], [2], *security group* is the primary means for cloud tenants to configure security policies to protect their virtual machine (VM) instances against attacks [3], [4]. Although similar to the conventional network firewalls in many ways, security groups have several distinct features that make their configuration somewhat more complex and trickier to use (see §II for details). Unlike firewalls where rules are typically configured by experienced network administrators, security groups and their constituent security rules must be specified by cloud tenants, some of whom may not be well-trained or lack an adequate network management background to properly configure security groups. Unfortunately, vulnerabilities in one tenant's VMs pose security threats not only to the tenant itself but also to the entire multi-tenant cloud platform. Ensuring that each cloud tenant properly specifies his/her security groups and the rules therein is therefore paramount to multi-tenant cloud platform providers.

In this paper we first conduct a measurement-oriented analysis of security group configuration and usage by tenants in a public IaaS cloud based on *real-world* datasets. Our goal of this measurement study is multi-fold: to understand what are the usage patterns ("good" and "bad" practices) in how cloud tenants configure their security groups, what they attempt to achieve, what are the common issues and potential security vulnerabilities, and how to help cloud tenants refine their security group configurations to prevent these issues and vulnerabilities. As an example of "bad" practices and potential vulnerabilities revealed by our analysis of a multi-tenant IaaS cloud system security group dataset, we find that a number of tenants simply allow all traffic (`0.0.0.0/0`) from both the external Internet and within the cloud to access their VMs. In general many tenants inappropriately configure their security groups by using loose, and sometimes inconsistent, rules (see § II and §V for more discussion on these and related points).

Motivated by the results and insights obtained from this measurement study, we propose and develop a cloud security group analysis tool called *Socrates*. Socrates takes the security group settings of each tenant, the VM mapping as well as the observed traffic flows (both *allowed* and *denied*) as inputs, and employs visual analytics to assist cloud tenants in understanding the static and dynamic *access relations* among VMs based on the security groups they have specified and the traffic observed. Furthermore, our tool also helps cloud tenants diagnose potential misconfigurations and provides suggestions to refine security group configurations based on real traffic traversing the tenant VMs. As a result, cloud tenants can view their security group configurations in a high-level, visualized manner, and revise their security group settings immediately after they realize some configurations do not meet their intent.

By applying *Socrates* to all existing tenants hosted on our IaaS cloud using the *week-long* datasets, we report some key results and lessons we have learned in §V. As alluded earlier, security groups are often set up by tenants who are "ordinary" application developers and may not be experts in network security. Hence we expect to see many configuration errors. Nonetheless we are surprised to find many configuration issues, some of which can lead to potential security vulnerabilities. For example, we find that more than 80% tenants configure security groups in a loose manner. In contrast, some tenants verbosely set rules leading to giant security groups with hundreds of rules. While many tenants create multiple security groups for their VMs, a large number of them do not have a clearly defined *structure* in mind when creating these security groups. *Socrates* also reveals many *redundant* or *inconsistent* rules in the security group configurations, likely the result of tenants' lack of knowledge about the intricacies of security groups (*e.g.*, rule ordering is immaterial) or mistakes in configuring rules. To the best of our knowledge, this is the

first work of analyzing cloud security groups. Our work sheds light on understanding the common usage for security groups and proposes a tool to better understand, diagnose and refine security group configurations.

II. Overview and Datasets

In this section, we first describe the basic concepts of IaaS cloud security groups and then the datasets used in our study.

IaaS Cloud: VMs and Security Groups. Creating a cloud application in an IaaS cloud starts with launching VM instances. One critical step in launching a VM is to configure security groups. A security group is a container for a set of security group rules. It provides tenants the ability to specify the type and direction of traffic allowed by VMs. Security groups are applied to individual VMs, whose private IP addresses are dynamically assigned only at the time they are launched – in other words, such private IP addresses are, in general, unbeknownst to the tenant at the time he/she specifies the security group rules. Unlike conventional firewall rules, the default action of security group rules is *deny*; thus, a tenant needs to explicitly specify what type of traffic (in terms of protocol and port) and from where (*e.g.*, in the form of a public or private IP address prefix) can access his/her instances. Furthermore, security groups can be "nested" in the sense that the security group rules in one security group, say, `SG-A`, can use the name of another security group (either belonging to the same tenant or another tenant), say, `SG-B` – in lieu of a (public or private) IP address prefix – to explicitly specify that the traffic from VMs in `SG-B` can access VMs in `SG-A` on ports permitted by the security group rules. Furthermore, the ordering of rules within a security group is immaterial; security group rules are not prioritized as in the case of firewall rules. Therefore, the most permissive rule gets applied if more than one rule is created for a specific port or IP range. Table I shows an example of a security group. Due to nested security group rules or IP ranges' coverage on VMs, there are *dependencies* among various security groups defined by one tenant (and sometimes among multiple tenants). Ideally, a tenant should create security groups based on the roles of VMs in a cloud service he/she develops.

Before getting launched, each VM must be assigned with at least one security group. A *default* security group is defined for all tenants, which by default denies all ingress traffic and allows all egress traffic and the traffic among the VMs associated with the *default* security group. When a VM is launched, it is associated with the *default* security group if no security group is specified by the tenant. In addition, a tenant can define and customize new security groups. One VM can be associated with multiple security groups, and one security group can be assigned to a collect of VMs. Therefore, one tenant can have a set of security groups and VMs, and the mapping between them can be fairly complex. Finally, tenants can configure security groups by adding or deleting rules, but not modifying an existing rule (*A rule cannot be modified once it is created*). Changes are automatically applied to the running VMs associated with the security group.

TABLE I: An example of security group with 3 rules.

Action	Protocol	Port Range	IP Range
ALLOW	TCP	80 – 5666	10.0.10.0/24
ALLOW	UDP	68 – 68	SG-A
ALLOW	ICMP	8,0	11.22.33.44/32

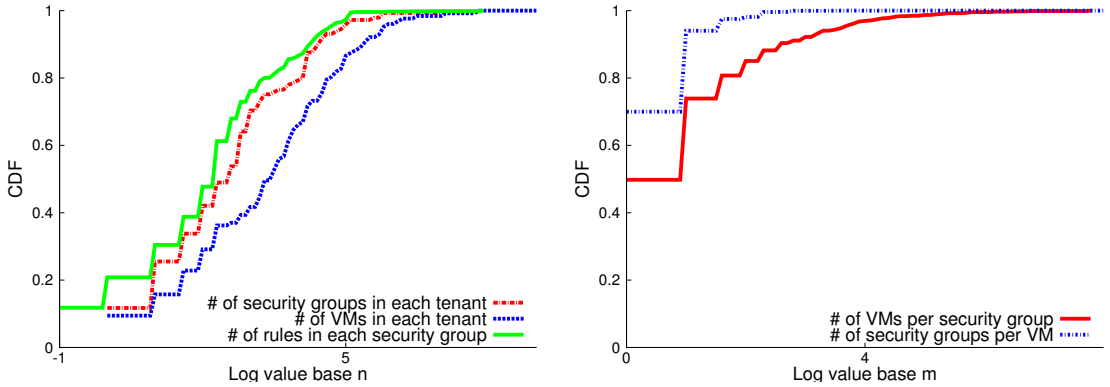
Datasets. The datasets used in our study are collected from a single multi-tenant data center running the OpenStack cloud software. There are three types of datasets: the `secgroup` dataset, the `VM-layout` dataset and the `sFlow` dataset. The first type of dataset is called `secgroup` which contains security groups and the constituent rules defined by cloud tenants. It contains five main fields: tenant ID, security group name, protocol type (TCP, UDP, or ICMP), port range (or ICMP type and code), and the source (IP range in the CIDR notation or the name of a security group). A tenant ID allows us to match the tenant across multiple datasets. The second type of dataset is the `VM-layout` that stores information about running VMs in the cloud at any given time. The important fields are VM name, tenant ID, associated security group(s), public IP address (if assigned), and private IP address. Both the security group and VM layout datasets are collected from the cloud configuration database. The last type of dataset is `sFlow` that contains flow traces (both *allowed* and *denied* flows) collected at each switch by random sampling. It stores packet header information, including source and destination IP addresses, TCP/UDP port numbers, time, switch identifier, and source/destination switch ports associated with the packet.

III. Current Usage of Security Groups

As *security group* is still a relatively unknown concept to many IaaS cloud customers, we first conduct an extensive measurement-based analysis of security group configuration and usage by tenants in a public IaaS cloud based on *real-world* datasets. In the following, we present some basic statistics and a few key results from this measurement-based analysis of the multi-tenant IaaS cloud security group, VM and flow datasets. The goal is to identify the common usage patterns in how cloud tenants generally configure their security groups. We also briefly point out a few "bad" practices in cloud tenant security group configurations, which we will expand on further in Section V in conjunction with the discussion of the results obtained from applying our *Socrates* tool.

A. Basic Statistics

Fig. 1a shows the number of security groups and the number of VMs in each tenant, as well as the number of rules that each security group has. The x-axis is the normalized value where n is a base value. As the results show, around 10% tenants have only one security group, and the remaining have at least two security groups. Most tenants have less than several dozen security groups, whereas not every security group plays a different role. The number of rules in security groups (log value) starts with -1 (it could be any negative value, and we use -1 for simplicity) at x-axis, because some tenants have empty security groups that do not have any rule. Apart from



(a) The number of security groups and VMs in each tenant, and the number of rules in each security group.

(b) The number of security groups associated with each VM and the number of VMs associated with each security group.

Fig. 1: Basic statistics of security group usage by tenants.

15% no-rule security groups, most security groups have less than one hundred rules.

Given the tenants that have multiple security groups and multiple VMs, we are interested in the association between security groups and VMs (shown in Fig. 1b). Our results show 50% security groups are associated with only one VM. In the rest half security groups, most of them are associated with a few dozen VMs, and very few of them are associated with more than half hundred VMs. 70% VMs are assigned with only one security group, and others are assigned with multiple security groups.

As depicted in Fig 2a, generally the more VMs a tenant has, the more security groups it tends to have, so the more sophisticated system the tenant is expected to build. However, we also notice that some tenants have a large number of VMs but only contain a few number of security groups. One reason is that these tenants have simple architectures but very high workload so that they need to launch a number of VMs to balance the workload. Another possible reason is that the tenants glue all rules in a few security groups instead of reasonably separating them into more security groups (discussed in Section V).

B. Rules in Security Groups

To investigate how security groups are configured in tenants, we start from studying their rules. Each rule consists of port and IP range. Based on the IP range, a rule can be classified into three groups: only accepting the external traffic[1], only accepting the internal traffic, accepting both external and internal traffic (*e.g.*, 0.0.0.0/0). As a security group is a set of rules, we can further determine whether a security group is: accepting only the external traffic, accepting only the internal traffic, or both. In our `secgroup` dataset, we find that 42% rules allow external traffic (referred to as *external rules*) and they are distributed in 34% security groups. 39% rules allow internal traffic and they are distributed in 61% security groups (referred to as *internal rules*). 19% rules allow traffic from everywhere (0.0.0.0/0) and they are distributed in 50% security groups (also referred to as *external rules*). In addition, a rule can be very restrictive or very permissive by setting the decimal in CIDR notation. For example, decimal 32 is used for specify an individual IP address, and decimal 0 means to cover all IP addresses. We find that 34% rules use decimal 32 (*e.g.*, a.b.c.d/32). Around 60% external rules use decimal 32 to set individual IP addresses, while most internal rules use IP blocks (*i.e.*, $0 <$ decimal < 32).

In terms of the port range used by each rule, our results show that the top five mostly-used TCP port ranges are 80, 443, 8080, 22, and 1-65535. We are surprised to see many rules set 1-65535 in port range, because simply allowing all ports is very risky. Moreover, ICMP rules' configurations are more biased, more than 90% ICMP rules are coarsely set to allow all types and all codes.

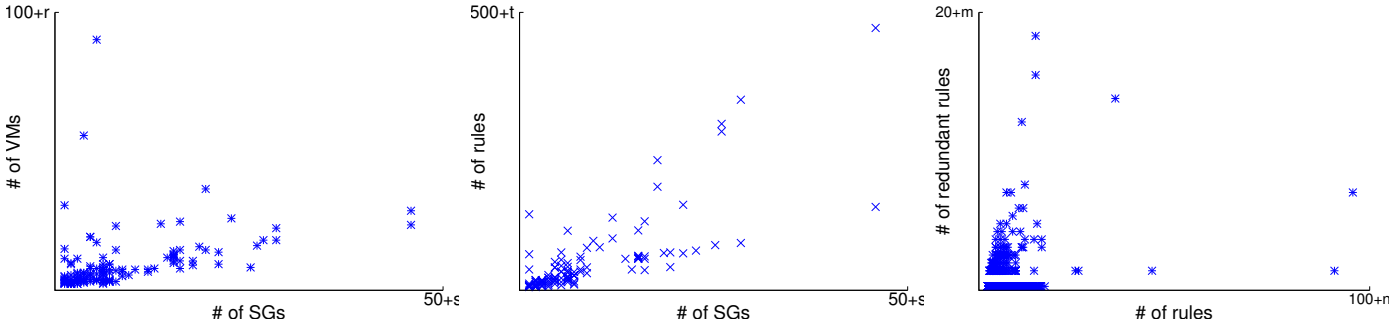
Furthermore, we also observe that 14% security groups distributed in 48% tenants contain *redundant* or *inconsistent* rules: for instance, two rules allow traffic on the same port (say, TCP 443) but from two different IP address prefixes, one a sub-prefix of the other. Such rules make little sense, as traffic will be allowed by the less restrictive rule. This appears to be a result of a tenant attempting to modify an existing rule by adding a new rule but forgetting to delete the old rule. Fig. 2c shows the number of rules and the number of redundant rules each security group has.

C. Security Group Dependency

Based on the understanding of rule settings and the fact that a security group is actually a set of rules, now we study the security group usage in tenant level. As a rule can be categorized into external rules and internal rules, a security group can also be categorized into *external* (only has external rules), *internal* (only has internal rules), and *mixed* (has both external and internal rules).

In our dataset, all tenants allow external traffic to some extent. 15% tenants consist of only external security groups. The security group rules for external traffic should be more

[1]We define *external IPs* as the addresses that do not belong to the IaaS cloud. In contrast, *internal IPs* are owned by the cloud. For simplicity, external traffic is referred to as the traffic between internal IPs and external IPs, and internal traffic denotes traffic between internal IPs.



(a) The relation between security groups and VMs. (b) The relation between security groups and rules. (c) The distribution of rules in every security group.

Fig. 2: The relation between security groups and VMs, and between security groups and rules.

carefully configured in order to protect the VMs from outside attacks. As most tenants have multiple security groups, we are interested in the relation among the security groups in the same tenant. The relation can be depicted as a graph (discussed in details in Section IV), where each security group is a node and each directed edge indicates the successor allows certain type of traffic from the predecessor. 70% security group graphs have bidirectional edges between each pair of security groups. Among them, around 40% share same port ranges on the same pair of bidirectional edges.

D. Bad Practice in Security Group Configurations

As part of the motivation for the *Socrates* tool, we provide some sample results from an initial analysis of the `secgroup` dataset (see Table II). Our analysis shows that "good practice" (*i.e.*, use *nested* security groups to scope communications among VMs) is not widely adopted yet – only 5% tenants employ nested security group rules. It reveals a fact that many cloud tenants have not completely grasped the concept of security groups or the subtle intricacies involved, and as a result, often specify rules that are either semantically incorrect or too loose.

We find that 24% of tenants open all ports on their VMs to accept traffic. Out of these tenants, 19% tenants allow traffic from 0.0.0.0/0, *i.e.*, accept traffic from anywhere on the Internet. This extremely-permissive setting exposes the tenants as victims of potential security attacks because it does not filter any traffic. When looking into the IP ranges specified in the rules, we find that some tenants do not even understand the CIDR notation. 13% of tenants in our dataset have rules with `a.b.c.d/0` (where a.b.c.d != 0.0.0.0) and 5% have rules with 0.0.0.0/x (where x!= 0), which is semantically incorrect. In addition, many tenants often use rules with 10.0.0.0/8 instead of *nested* security groups when their intention is to simply enable communications among VMs between certain security groups (see Section V for more detail).

In some tenants' configurations, all of their security groups surprisingly open all ports for all VMs belonging to the tenants. This loose setting arouse our investigation in their flow usage. We find that their flows are much more restrictive (*i.e.*, only contacting some ports from a subset of VMs) compared to the configured rules. These observations motivate us to design and develop a tool which visualizes the security group setting, analyzes real flows against the security group rules, and generates diagnostic reports, which detailing problems with the security group rules. Section IV explains the design of our tool *Socrates*.

TABLE II: Initial analysis of `secgroup` dataset.

Usage	Tenants	Rules
Bad usage	24%	Open all ports (1–65535)
Bad usage	13%	Meaningless CIDR: a.b.c.d/0 (a.b.c.d != 0.0.0.0)
Bad usage	5%	Meaningless CIDR: 0.0.0.0/x (x!=0)
Good usage	5%	Use nested security groups

IV. *Socrates*: A Security Group Analysis Tool

In this section we provide an overview of *Socrates* – a cloud security group analysis tool that we have developed [2] – and briefly describe its key components. Part of the rationale for *Socrates* is our recognition that many IaaS cloud tenants are "ordinary" application developers who may not be very familiar with notion of security group and its intricacies, let alone being a network security expert. Ideally, when a tenant develops and deploys a service or application an IaaS cloud platform, security groups should be created to reflect the roles of VMs and meet their security and management requirements. As we briefly discussed in Section III and further expanded on in Section V, creating and configuring security groups can be quite a challenging task for many tenants. Unfortunately, vulnerabilities in one tenant's VMs pose security threats not only to other tenants but also to the entire multi-tenant cloud platform. Hence ensuring security for each tenant is crucial.

Socrates is designed to assist cloud tenants in understanding their security group settings and help them diagnose their configuration issues. *Socrates* takes the security group settings of each tenant, the VM mapping as well as the observed traffic flows (both *allowed* and *denied*) as inputs, and produces a visual representation of *security group/VM structure* as well as a *diagnosis and recommendation report* to help tenants diagnose and improve their security group configurations based

[2]The name, *Socrates*, is derived as an anagram of the capitalized letters in *SECurity gROup AnalySis Tool*. An initial design of the tool [5], together with some preliminary results, have been reported in a *short paper*.

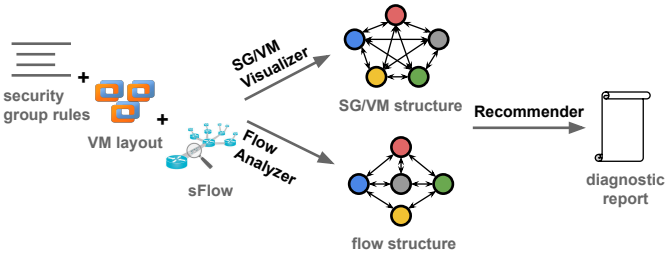


Fig. 3: *Socrates* workflow.

on observed network traffic. *Socrates* consists of three key components: *visualizer*, *flow analyzer*, and *recommender*, see Fig. 3 for a schematic illustration.

Security Group/VM Structure Visualizer: It displays the dependencies of security groups and VMs through directed graph representations based on the (static) security group settings and the (dynamic) VMs to security group mappings. The dependency between security groups reveals the cloud service infrastructure design that a customer has envisioned. Hence, a directed graph (referred to as a *security group structure graph*) is generated to represent security groups of one tenant, where nodes stand for individual security groups and the edges encode dependencies between security groups. Each directed edge indicates the successor security group allows the traffic satisfying the specific port range and IP range from the predecessor security group (or external networks). From the graph, we further identify *tiers* to which security groups belong. A security group is defined as tier N if and only if it allows traffic from tier $N-1$ but not from any other lower tiers. For example, *tier 1* security groups contain at least one rule explicitly allowing external traffic. *Tier 2* security groups allow traffic from *tier 1* security groups but not from external networks. After building the security group structure graph, we next add the VM-level structure into the graph by mapping VMs to assigned security groups. VMs are displayed as rectangular nodes inside the corresponding security groups. In addition, we introduce edges between VMs within the same security group to indicate that traffic is allowed between a particular pair of VMs. On the other hand, the dependency between VMs across two security groups are already captured by edges between security groups. Fig. 4a depicts the security group/VM structure for a real tenant from our datasets, nicknamed "Alice", where all security groups belong to tier 1 since they all allow external traffic.

Flow Analyzer: It infers the cloud service infrastructure design by analyzing the traffic flows associated with the service, both *allowed* and *blocked*. A particular flow between a source VM and a destination VM is considered *allowed* or *blocked* based on whether it is allowed by rules in the destination VM security group or not. To build the flow structure, the analyzer marks flows as either *allowed* or *blocked* by checking each flow with the rules of all the associated security groups. With both allowed and blocked flows, we build the flow structure, a directed graph at the VM-level, based on flows' src_IPs, dst_IPs and dst_ports. The directed edges are labeled as "allow" or "block" to differentiate the flows are accepted by rules or not. This VM-level graph can also be easily converted to a security group level graph by aggregating the flows of VMs belonging to the same security group. An example of flow structure for tenant Alice is shown in Fig. 4b, where we see that the (dynamic) flow structure is more "sophisticated", *e.g.*, containing more "tiers", than the simple tier-1 structure depicted in Fig. 4a.

Recommender. It utilizes the information generated by the security group structure and flow structure in order to identify the differences between the rules created and the flows accepted or denied by customer VMs. It further alters customers about the mismatch as as well offers suggestions to modify security group by providing the analysis report[3]. If the security groups are defined too widely, we can recommend that tenants refine their security groups to restrict ports and IPs that do not appear in the flow structure. For example, given most security group and VM structures are complete graphs, the flow structure can show more sophisticated structures. It also analyzes the causes of blocked flows. In terms of the "block" edges, if the same kind (same src_IP, dst_IP and dst_port) of blocked flows keeps coming for a long time, *Socrates* raises alert to customers in case of potential misconfigurations or attempt of attacks.

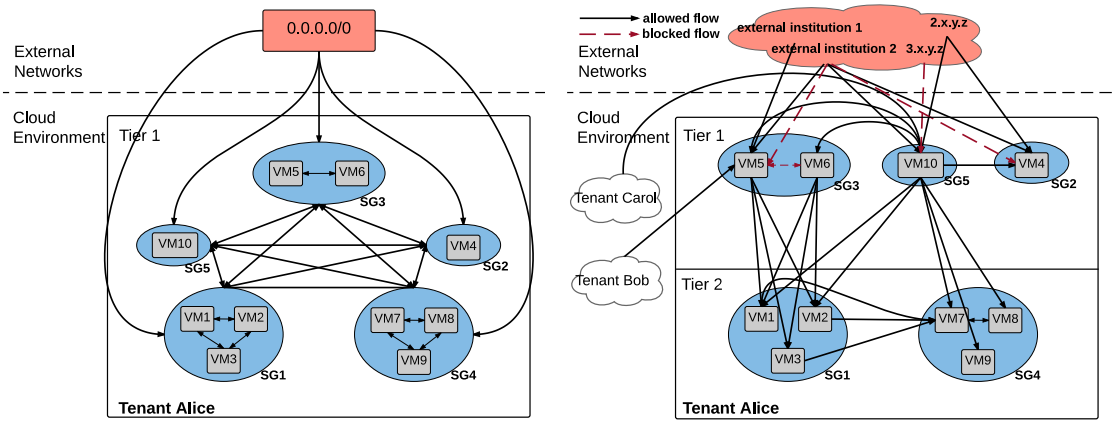
V. Security Group Configuration Analysis and Diagnosis

To evaluate the efficacy of the proposed tool, we apply *Socrates* to examine and analyze the security group configuration issues of all tenants on our IaaS cloud, using *one-week* datasets of tenant security group settings, VM layouts and traffic flows. In the following we will first provide a brief overview of the results we have obtained, highlighting a few configuration issues uncovered by *Socrates*. Then we will discuss about *structural analysis* of security group configurations to illustrate how *Socrates* can help tenants visually analyze their security group settings and track their changes over time. We will also present analysis and discussion of the uncovered configuration issues in the end.

A. A Brief Overview of Results Obtained via Socrates

As alluded earlier, in contrast to firewall rules which tend to be configured by professional network operators, security groups are often set up by tenants who are "ordinary" application developers who may not be an expert in network security. Hence we expect to see many configuration errors. Nonetheless we are surprised to find that around 50% tenants have at least one security group without any rule configured. A few of them even have VMs assigned to these empty security groups. As revealed by the flow analysis, many tenants configure rules *loosely*, for example, using rules with sources such as 0.0.0.0/0 or 10.0.0.0/8, without regards to the

[3]We quantify mismatches using the Jaccard distances of corresponding IP ranges and port ranges within two structures. While the threshold on Jaccard distances can be set according to management needs, we choose a conservative value of 0.1 in our experiments. In other words, we only study most significant mismatches.



(a) Security Group/VM structure of tenant Alice.

(b) Flow structure for tenant Alice.

Fig. 4: Examples of SG/VM Structure and Flow Structure.

actual application requirements. Other tenants configure rules *verbosely*, *e.g.*, by creating one rule per VM (*i.e.*, using a /32 IP address as the source), which leads to a giant security group with many rules. While many tenants create multiple security groups for their VMs, a large number of them do not seem to have a clearly defined *structure* in mind when creating these security groups (see Section V-B). Very few leverage (*nested*) security group names as an effective way to permit only traffic between VMs of specific security groups and restrict traffic from other VMs not belonging to these security groups; instead they often resort to either using overly permissive rules with 10.0.0.0/8 or 10.0.0.0/24 or creating one rule per VM address as stated earlier. *Socrates* also reveals many *redundant* or *inconsistent* rules in the security group configurations, likely the result of tenants' lack of knowledge about the intricacies of security groups (*e.g.*, rule ordering is immaterial) or mistakes in configuring rules.

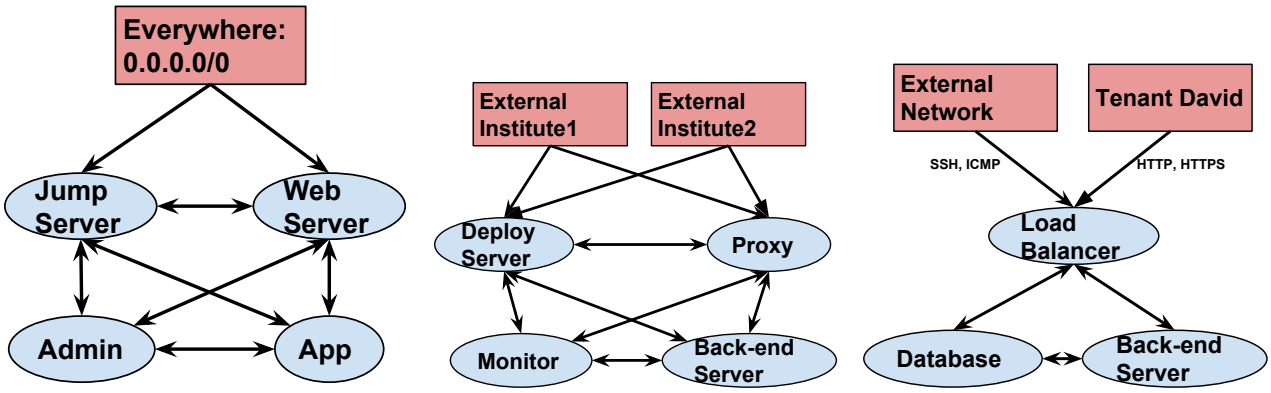
B. Structural Analysis of Security Group Configurations

Socrates takes the security group settings of each tenant, the VM mapping as well as the observed traffic flows (both allowed and denied) as inputs, and employs visual analytics to assist cloud tenants in understanding the static and dynamic access relations among VMs based on the security groups they have specified and the traffic observed. In this section we report some key results we have obtained by applying *Socrates* to all tenants' security group settings using the one-week datasets.

The goal of *structural analysis* of security group configurations is to help tenants visualize and understand the relations among various security groups they have configured, whether they reflect the roles and application requirements of the VMs associated with these security groups, and how the observed traffic (both *allowed* and *blocked*) traffic match what the security group rules are intended to accomplish. We find that although a majority of tenants have more than one security group configured, many do not appear to have a clearly defined structure in mind. We observe that 51% tenants tend to have a single tier, whereas the remaining have two tiers. No tenant has more than two tiers, despite some of them have configured a large number of rules that apply to a large number of VMs.

Fig. 5 depicts three *representative* examples of *two-tiered* security group structures generated by *Socrates*, which we classify them as: (i) *public customer facing web service*, (ii) *private enterprise application*, and (iii) *back-end service support*. The tenants in category (i) use the IaaS cloud platform to deploy a public web service serving customers from everywhere (0.0.0.0/0), while the tenants in category (ii) may have likely migrated a private enterprise application to the IaaS cloud platform and thus restrict it to a specific set of IP address ranges belonging to the private enterprises. The tenants in category (iii) on the other hand leverage the the IaaS cloud platform for back-end service (*e.g.*, databases) support for another service (or tenant). In this case, we often see that traffic from another tenant (often in category (i)) is allowed. Judging based on the names of the tenants involved, the two tenants likely belong to the same owner. In category (iii), although some traffic from one or two external networks are allowed, they are primarily for the management purpose (SSH or ping from the external networks). The remaining rules are all restricted to internal VMs, and the commonly used ports are for web proxy services, databases services, synchronization services, and monitoring services. For tenants with two tiers, 61% are public customer facing, 32% tenants are private enterprise application, and 7% tenant are back-end service support.

The (static) structure of the security group settings is also reflected by the *dynamic structure* in the observed traffic flows through the flow analysis. We find that VMs associated with the tier-1 security groups often function as web servers/web proxies, load balancers, or jump servers. VMs associated with many tier 2 security groups are running database services, certain application services or monitoring services. In particular, we notice that VMs associated with "monitoring" security groups only send traffic to other VMs, but hardly allow traffic from other VMs.



(a) Public Customer Facing Tenant. (b) Private Enterprise Application Tenant. (c) Back-end Service Support Tenant.

Fig. 5: Three categories of tenant structures.

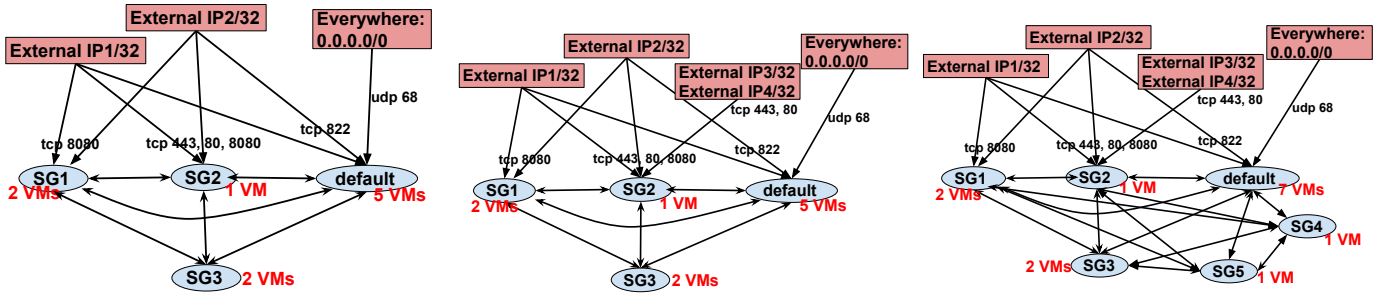
Potential Vulnerabilities. As stated earlier, we find that many tenants have only a single-tier structure. Further analysis reveals that for a majority of tenants (70%), their security groups form a *full mesh*, *i.e.*, any pair of security groups are allowed to communicate with each other. Based on our observation, the existence of many full meshes is caused by tenants extensively using 10.0.0.0/8 and 10.x.x.x/24 to grant access to their VMs. In particular, we find that 16% of tenants use 10.0.0.0/8, 23% of tenants use 10.x.x.x/24, and 44% of tenants use 10.x.x.x/y where $8 < y < 24$. On the other hand, based on the analysis of observed traffic flows of these tenants, these rules are meant to apply to VMs belonging to the tenants' own security groups. These overly permissive rules imply that any other VMs in the cloud platform (even those not belonging to the tenants) are allowed to access these VMs, thereby creating potential security vulnerabilities. As a tenant may not know the private IP address range dynamically assigned to its VMs, many resort to simply use 10.x.x.x/8 or 10.x.x.x/24 to cover its VMs, as opposed to use the names of its security groups directly. A particularly concerning problem with these tenants with such a "full-mesh" single tier structure is that as some of the VMs are associated with security groups which are "customer-facing", *i.e.*, allowing external traffic to access them, one compromised customer-facing VM can lead to other VMs (even though they are not assigned any public IP address, thus not directly addressable from the outside world) being potentially compromised. By analyzing both the *static* security group settings and dynamic VM layouts and traffic flow structures, *Socrates* is capable to alert tenants about such potential security vulnerabilities and suggest alternative security group structures based on the common traffic patterns observed among VMs.

C. Tracking Configuration Changes

By applying *Socrates* to the security group settings, VM layouts and flow datasets over one week, we also track how tenants modify the security group rules to experiment with and refine their settings to meet application needs, or adapt to changing application requirements. By observing what flows are allowed and what are blocked, and how they vary over a period of one week, we can also get a sense of what are "normal" traffic activities, but what may be "anomalous" traffic activities.

In our datasets, 14% of the tenants made security groups configuration changes in the one week period. Some tenants made many changes, such as adding new security groups, deleting existing security groups. Other tenants made slight modifications to existing security groups by either adding new rules (*e.g.*, open more ports or allow more IPs) or deleting existing rules. In addition, some new VMs were launched with newly-added security groups, while some existing VMs were terminated with removing existing security groups. We observe that among the tenants which generate most traffic (top 11% tenants), their security group configurations hardly change at all over the one week period, although the numbers of VMs launched and the amount of flows may vary over time. This observation indicates that the services operated by these top tenants are well-developed and running in a stable mode. In contrast, we find that a few tenants with quite less traffic frequently changed their security group configurations and VM association over the one week period, suggesting that they were still developing their services and were experimenting with the security group settings.

Fig. 6 provides an example where a tenant Eric modifies its security groups in the one week period. Initially (see Fig. 6a), the tenant has four security groups and five VMs. The number beside each security group indicates the number of VMs associated with it. Note here all VMs are also associated with the *default* security group. Except SG3, the other security groups allow external traffic, so that they are in Tier 1. After half a day (Fig. 6b), additional rules are added to SG2 to allow HTTP and HTTPS traffic from more external IPs. By analyzing the observed flows of this tenant, we see traffic from these newly-allowed external IP addresses in the same hour as the rules were added. Several days later (Fig. 6c), two new security groups, SG4 and SG5, were added, with rules allowing traffic from other security groups. Similar to SG3, these two new security groups function as back-end application services, but with different ports open. Two new VMs were



(a) (b) (c)

Fig. 6: Snapshots of an actively-developing tenant Eric.

launched, one associated with SG4 and one with SG5. The flow datasets reveal that indeed there is traffic between the two VMs.

This example helps illustrate that when a tenant modifies its security group settings, its intention is often to permit or restrict certain traffic. Therefore, the dynamic structure in the observed flows should also change accordingly. However, we have also observed that the dynamic flow structures change before the security group configuration is modified. While flow structures change may be due to, *e.g.*, attacks, when such changes persist over time, they can be an indication of changing application requirements or a change in the nature of services. For example, if the same type of flows continuously get blocked for a long time, this may be due to a "misconfiguration" (a previously too restrictive rule may need to be relaxed). In this case, our tool will raise a red flag to notify the tenant.

Potential Vulnerabilities. As tenants add new rules or modify their existing security group settings over time to meet changing application or service requirements, many forget to delete their old rules. These lead to *redundant* or *inconsistent* rules in the security group configurations, say, with multiple rules apply to the same or overlapping or a subset of IP address blocks which permit traffic on a different set of TCP/UDP ports. Some of these configuration issues may be due to tenants' lack of knowledge in security group configurations: they may not realize that once a rule is set, it cannot be modified/updated; creating a new rule, say, applies to the same IP prefix block but with a new port range, does not invalidate the previously configured rule – old rules must be explicitly deleted when they are no longer needed. Some tenants may simply forget to delete old rules when creating new rules or forget about the existence of these old rules. Given that the ordering of rules in a security group does not matter, such mistakes can potentially create security holes, especially when a new rule is put in place to limit certain unwanted traffic that an old rule previously allows. *Socrates* is able to explicitly flag such redundant or inconsistent rules and alert the tenants about such configuration issues which potentially create security vulnerabilities.

D. Loose, Verbose, and Inconsistent Configurations

As mentioned earlier, it is surprising that most tenants (more than 80%) set security groups in a loose manner. Tenants are suggested to restrict IP ranges to credible IP blocks by using proper CIDR notation or security group names. In addition, tenants are encouraged to use nested security groups to specify IP ranges. This feature enables allowing traffic from all VMs associated with the nested security group without using individual IPs or IP ranges. If there is any VM newly-launched or stopped, the tenant does not need to modify the rules. Based on our observation, the flow structure often time reveals a subset of the access relationship than the security group structure generated by security group settings. It also tends to reveal more about the tier structure. One of the key reasons is that tenants extensively set security groups loosely, such as 10.0.0.0/8 and 10.x.x.x/24. Hence, the corresponding security group settings can be refined to be more restrictive based on the flow structure. In addition to setting rules loosely, some tenants also set security groups loosely. Specifically, instead of setting security groups distinctly to present their roles, the tenants simply replicate security groups over and over again. In this case, these security groups have exactly the same rules but different security group names. However, by looking into their flow structures, we clearly see each of these security group's real intentions and functions are entirely different. Hence, we suggest the tenant should refine security groups to reflect their distinct roles.

In contrast to setting security groups loosely, a few tenants in our cloud set their security groups in an extremely verbose manner. Especially some tenants only have one giant security group with hundreds of rules. We observe that it is because the rules are set by using individual IPs of VMs. If there is any VM launched or stopped, the same type of rules need to be added or deleted.

Redundant or *inconsistent* rules are the multiple rules which apply to the same or a subset of IP address blocks/ports which permit traffic on a different set of ports/IP address blocks, one a subset of the other. Such rules make little sense, as traffic will be allowed by the most permissive rule. Among the tenants which have redundant rules, 30% tenants have more permissive rules followed by more restrictive rules, 40% tenants have

more restrictive rules followed by more permissive rules, and 30% tenants have both cases. With the analysis of sFlow dataset, in terms of the tenants which have more permissive rules coming first, 83% tenants have most flows allowed by the former permissive rule but cannot be allowed by the latter restrictive one. 17% tenants have most flows allowed by the former permissive rule and could also be allowed by the restrictive rule. In terms of the tenants which have more restrictive rules coming first, we find that 75% of them have only a few flows allowed by the former restrictive rule and most flows accepted by the later permissive rule, which indicates the customer intends to create a more permissive rule to replace the restrictive one, but unfortunately forgets to delete the restrictive rule. 25% tenants have most flows allowed by the former restrictive rule while only a few allowed by the latter permissive rule.

VI. Related Work

Security is a key concern in the adoption of cloud computing. To this end, researchers have developed many security solutions to be offered as a cloud service [6], [7], [8]. Brown et al. [6] developed trusted platform-as-a-service for cloud tenants to deploy applications in the cloud in a trustworthy manner. Hyun-wook et al. [7] offered virtual machine introspection (VMI) as a cloud service to allow customers to develop their own tamper-resistant security tools without relying on cloud providers. Srivastava et al. [9] created a notion of cloud app marketplace for distributing system and security services packaged in VMs.

Popular IaaS clouds [1], [2] provide customers with security groups [10], [3], [11], [12], [4] to ensure their VM instance security. Security groups act as firewalls which are sets of rules controlling the traffic for VMs. While there have been many prior studies on firewall analysis (see, *e.g.*, [13], [14], [15], [16], [17], [18], [19]), security groups differ from firewalls in many aspects. To the best of our knowledge, our work is the first to analyze security group usage in multi-tenant IaaS clouds. We apply our proposed security group analysis tool *Socrates* (its initial design was reported in a short workshop paper [5]) – to help cloud tenants visualize and understand the access relations among VM instances based on the specified security groups and the traffic observed and to further diagnose potential misconfigurations.

VII. Conclusions

The contributions of our paper are summarized below: i) Using the real-world datasets from a public multi-tenant IaaS cloud, we have conducted a first measurement-based analysis of security group configuration and usage. Through this measurement-based analysis, we have studied the common usage patterns in how cloud tenants generally configure their security groups. We revealed some issues and potential vulnerabilities in cloud tenant security group configurations. ii) Motivated by the results and insights obtained from this measurement study, we then proposed and developed a security group analysis tool called *Socrates*. *Socrates* enables tenants visualize and hence to understand the static and dynamic access relations among VMs. *Socrates* also helps diagnose potential misconfigurations and provides suggestions to refine security group configurations based on real traffic traversing tenants VMs. iii) We have applied *Socrates* on all tenants hosted on the IaaS public cloud and demonstrate its effectiveness in helping cloud tenants analyze, visual, diagnose and refine their security group settings. To the best of our knowledge, we believe that our work is the first to analyze cloud security group usage based on real-world datasets, and to develop a tool to help cloud tenants to understand, diagnose and better refine their security group configurations. Our work sheds light on the common usage ("good" and "bad" practices) of cloud security groups and on how to design better and more secure cloud systems and services.

Acknowledgment. We thank the anonymous reviews for their valuable feedback. This research was supported in part by NSF grants CNS-1117536, CRI-1305237, CNS-1411636 and DTRA grant HDTRA1-14-1-0040 and DoD ARO MURI Award W911NF-12-1-0385.

References

[1] "Amazon EC2," *http://aws.amazon.com/ec2/*.
[2] "OpenStack," *http://www.openstack.org/*.
[3] "AWS EC2 security groups," *http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html*.
[4] "OpenStack security groups," *http://docs.openstack.org/network-admin/admin/content/securitygroups.html*.
[5] C. Jin, A. Srivastava, Y. Jin, and Z.-L. Zhang, "Secgras: Security group analysis as a cloud service," in *IEEE ICNP*, October 2014.
[6] A. Brown and J. Chase, "Trusted platform-as-a-service: A foundation for trustworthy cloud-hosted applications," in *ACM Cloud Computing Security Workshop*, 2011.
[7] H. wook Baek, A. Srivastava, and J. K. V. der Merwe, "Cloudvmi: Virtual machine introspection as a cloud service," in *IEEE International Conference on Cloud Engineering (IC2E)*, 2014.
[8] A. Srivastava, H. Raj, J. Giffin, and P. England, "Trusted VM Snapshots in Untrusted Cloud Infrastructures," in *Proceedings of the 15th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2012.
[9] A. Srivastava and V. Ganapathy, "Towards a richer model for cloud app markets," in *ACM Cloud Computing Security Workshop*, 2012.
[10] "Amazon web services:overview of security processes," *http://aws.amazon.com/security/*.
[11] "AWS security groups for VPC," *http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_SecurityGroups.html*.
[12] "OpenStack API for security groups and rules," *http://docs.openstack.org/api/openstack-network/2.0/content/security-groups-ext.html*.
[13] E. Al-Shaer and H. Hamed, "Firewall policy advisor for anomaly detection and rule editing," in *Proc. IEEE/IFIP Integrated Management Conference*, March 2003.
[14] ——, "Discovery of policy anomalies in distributed firewalls," in *Proc. IEEE Infocomm*, March 2004.
[15] S. M. Bellovin, "Distributed firewalls," in *;login:*, November 1999.
[16] M. J. Chapple, J. DArcy, and A. Striegel, "An analysis of firewallrulebase (mis)management practices," in *ISSA*, February 2009.
[17] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith, "Implementing a distributed firewall," in *CCS*, November 2000.
[18] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra, "Fireman: a toolkit for firewall modeling and analysis," in *IEEE Symposium on Security and Privacy*, May 2006.
[19] K. Golnabi, R. K. Min, L. Khan, and E. Al-Shaer, "Analysis of firewall policy rules using data mining techniques," in *IEEE/IFIP Network Operations and Management Symposium*, April 2006.