Towards Automated Identification of Security Zone Classification in Enterprise Networks

HariGovind V. Ramasamy†, Cheng-Lin Tsao‡, Birgit Pfitzmann†,
Nikolai Joukov†, and James W. Murray§

†*Services Research, IBM Research*
Hawthorne, NY, USA
{*hvramasa,bpfitzm,*
njoukov}*@us.ibm.com*

‡*Georgia Tech*
Atlanta, GA, USA
cltsao@gatech.edu

§*IBM*
Research Triangle Park
Raleigh, NC, USA
jmurray@us.ibm.com

Abstract

Knowledge of the security zone classification of devices in an enterprise information technology (IT) infrastructure is essential in many enterprise IT transformation and optimization activities. We describe a systematic and semi-automated approach for discovering the security zone classification of devices in an enterprise network. For reduced interference with normal operation of the IT infrastructure, our approach is structured in stages, each consisting of two phases: one phase involves collecting information about actually allowed network flows, followed by an analysis phase. As part of our approach, we describe an elimination-based inference algorithm. We also present an alternative to the algorithm based on the Constraint Satisfaction Problem, and explore trade-offs between the two. Using a case study, we demonstrate the validity of our approach.

1 Introduction

The network infrastructure of a modern enterprise is a complex system partitioned by enterprise firewalls into several logical network areas, called *security zones*. Informally, a security zone consists of one or more subnets. Each security zone belongs to a *zone classification* (or simply, *classification*), and consists of devices[1] subject to the same enterprise-level security requirements. The classifications in an enterprise network along with security requirements for systems and services within each classification, is usually documented in the security policy of the enterprise. The policy defines permitted flows between classifications.

Most enterprises have at least three classifications: intranet, extranet, and opennet (Figure 1). The intranet is a trusted network environment for hosting systems, services, and data internal to the enterprise. The opennet is an untrusted network environment (e.g., the Internet) that includes all systems external to the enterprise. Security zones belonging to the extranet classification are commonly referred to as demilitarized zones (DMZ), and serve as a buffer between zones belonging to the intranet and extranet classifications. Large enterprises often have more than three classifications. There may be special extranets to host services used jointly by partners, vendors, and suppliers. Zones hosting test and development services typically have classifications different from those hosting production services. Often, the intranet itself may consist of further classifications, depending on the sensitivity of the data resident and the business value supported by the constituent systems.

While most enterprises only have a small number of classifications, there may be a large number of security zones for each classification. The reason is that, despite what the name may suggest, security zones are not created solely for security purposes. Organizational, geographical, functional, and administrative factors also drive the creation of security zones. Even if they have the same classification, geographically distinct devices would have to be placed in different security zones. Within a given location, different organizational divisions may create and govern their own security zones. Within a given division, different business applications may be placed within their own zones. As a result, many enterprises have a sprawl of security zones. It is not uncommon for large enterprises to have hundreds or even thousands of security zones spanning multiple locations.

Knowledge of the zone classification of devices is a requirement in many enterprise IT transformation and optimization activities, such as desktop migration, cloud migration, firewall migration, and enterprise security refresh. The reason is that security zone classification provides important connectivity and isolation criteria that need to be upheld during and after such activities. For example, in server consolidation and virtualization activities, servers have to be migrated from a source environment to a target environment, and communication controls between servers belonging to different zones have to be reproduced in the target environment. Security zone classification of devices is also required for compliance and audit purposes, e.g., to assess whether end-to-end

[1] Unless specified, we use the term *device* in a broad sense to cover computing, network, and storage devices, both physical and virtual.

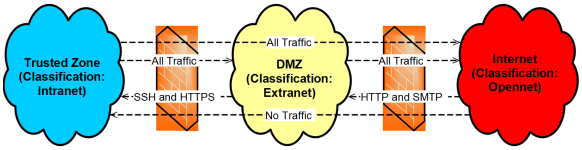


Figure 1: Security Zones and Zone Classifications in a Simplistic Enterprise

data flows are in compliance with enterprise policy.

From our experience in several client engagements, we have observed that many enterprises lack an inventory that details the zone classification of devices. Typically, when needed, such information is obtained by contacting system and network administrators. This manual approach often yields unreliable, outdated, and incomplete information. If a system or network administrator had moved on to a different company, no one in the enterprise may have security zone classification information for devices previously under his/her responsibility. Thus, there is a clear need for automated approaches to obtaining security zone information. The need is even greater when an external service provider or consulting organization which has no prior knowledge of or privileged access to the enterprise environment is called upon to perform enterprise IT transformation or optimization activities.

There are many network administration tools such as Nmap [6], traceroute, netstat, and SNMP-based approaches that can collect and analyze information about network configuration of devices in the infrastructure. Advanced tools can estimate the run-time network configuration of applications by analyzing traffic information provided by network services. There are also many academic and commercial firewall configuration analysis tools (e.g., [1, 2, 8]). However, these tools alone will not reveal what security zone classification (e.g., service provider extranet, customer extranet, or confidential intranet) a firewall interface or the devices placed behind that interface belong to.

In this paper, we describe a systematic and semi-automated approach for discovering security zone information in enterprise networks. Our approach is based on the observation that for reliably discovering security zone classification, information about network configuration alone is not sufficient. Since zones are created with enterprise security policies in mind, that information has to be compared against what is expected from a policy point of view. Our approach proceeds in stages, with each *stage* consisting of an information collection phase and an analysis phase. The collection phase obtains information about actually allowed network flows. In the analysis phase, an elimination-based inference algorithm may be executed. The algorithm eliminates classifications from an initial assignment of all possible classifications by comparing actually allowed network flows with flows permitted by enterprise security policy. We provide an alternative to the algorithm based on the Constraint Satisfaction Problem (CSP) [7]. We explore the trade-offs between the inference algorithm and the CSP-based solution, and outline how they can be used in concert for improved efficiency. We describe a case study to demonstrate the utility and validity of our approach.

2 Solution Approach

We distinguish between network flows *actually allowed* and those *permitted* by policy. There are several known methods for collecting information about actually allowed network flows. Examples include (i) configuration analysis of firewalls, hosts, and applications [8], (ii) analysis of network statistics and flow logs [3], (iii) active probes [6], and (iv) packet analysis [9]. Different collection methods involve varying interference with the normal network operation. Our approach sequences them in such a way that lower interference methods are executed before higher interference ones. The collection phases are interleaved with analysis phases whose results are used to reduce the scope of deployment of subsequent higher-interference collection methods. We call this strategy *incremental discovery*.

We now illustrate incremental discovery, based on two sample collection methods: `netstat` and connectivity probing. The `netstat` command, supported on most devices and OS platforms, is one way of collecting network configuration and run-time (network-related) application behavior. Execution of the command does not generate any external traffic. End-to-end connectivity probing can determine whether packets are able to reach a peer device through any intermediate filters. It can also identify open ports at the peer. Common applications and tools such as telnet, ftp, nslookup, and Nmap can be used for probing. Since the probe traffic may raise security concerns, proper coordination with administrators and users may be necessary when using this method.

In incremental discovery, `netstat`-based discovery is performed prior to connectivity probing. Flows already observed through netstat-based discovery can be skipped during connectivity probing. More importantly, analysis of `netstat` command output from various devices helps identify which devices belong to the same subnet. That result coupled with the observation that all devices in the same subnet have the same classification can be used to reduce the scope of probing; instead of probing device-to-device connectivity, probing subnet-to-subnet connectivity may suffice to determine the filters placed at the intermediate enterprise firewalls.

Each analysis phase may involve executing an inference algorithm which derives the security zone information by comparing enterprise security policy with information about actually allowed network flows. We de-

scribe our system model in Section 2.1, and present the inference algorithm in Section 2.2. In Section 2.3, we describe an alternative formulation of the zone classification problem based on the Constraint Satisfaction Problem (CSP). We illustrate how these two methods can be used in concert for improved efficiency.

2.1 System Model

Network area is an intermediate construct we use in the process of deriving security zones in a network environment and identifying their classifications. A network area may consist of a device or a grouping of logically adjacent devices (such as subnet). A security zone is composed of one or more subnets.

We use a unified framework based on the notion of feasibility sets [5] for representing both actually allowed network flows and those permitted by enterprise policy. A feasibility set $\mathcal{F}_{a_i->a_j} = \{x|f_{ij}(x)\}$ is used to denote the traffic flows from one network area a_i to another a_j. Guttman and Herzog [5] define it as the set of all abstract packets that survive all of the filters traversed along the path between a_i and a_j. Here, x is a packet and the predicate $f_{ij}(x)$ is defined over the fields of x. The predicate may include regular expression matches, denoted by $\cong$. A field y of packet x is denoted by $x.y$, a sub-field z of y is denoted by $x.y.z$, and so on. Typical packet fields that are considered include source IP address, source port, destination IP address, destination port, and protocol type. If $\mathcal{F}_{a_i->a_j}$ = `true`, then all flows are allowed from a_i to a_j. If $\mathcal{F}_{a_i->a_j}$ = `false`, then no flow is allowed from a_i to a_j.

Example 1. Suppose that a firewall with its inside interface on network area a_j is configured to allow all traffic from the 192.168.1.0/8 network, and only SSH and HTTPS traffic from all other networks. In this case, the feasibility set for traffic reaching a_j from any network area a_i, where $i \neq j$, is expressed as:
$\mathcal{F}_{a_i->a_j}$ = $\{x$ | (x.IP.SourceAddr $\cong$ 192.168.1.[0..255]) $\vee$ (x.IP.Protocol = 6 $\wedge$ x.TCP.DstPort $\in$ {22,443})$\}$. □

High-level security policies are usually documented in natural language or in a format intuitive to a human operator. The policy guides network, security, and system administrators to construct and maintain the infrastructure that complies with the security goal. Manual effort is needed to transform these policies into feasibility sets. However, since enterprise security policies are relatively static, the translation is typically a one-time effort.

Example 2. Consider an enterprise-level policy that permits only strongly authenticated traffic from the DMZ to the intranet. Strong authentication exists when a system or user can prove knowledge of a secret (e.g., password, private key) without it being observed or revealed. A lower-level policy may further enumerate the allowed set of strongly authenticated protocols as: (i) TCP traffic (i.e., IP Protocol 6) to ports 22 (SSH), 25 (SMTP), 389 (LDAP), and 443 (HTTPS), (ii) UDP traffic (i.e., IP Protocol 17) to port 500 (IPSec - Internet Key Exchange), and (ii) IPSec traffic (i.e., IP Protocols 50 and 51).

The feasibility set for the policy can be expressed as:
$\mathcal{F}_{a_i->a_j}$ = $\{x$ | (x.IP.Protocol = 6 $\wedge$ x.TCP.DstPort $\in$ {22,25,389,443}) $\vee$ (x.IP.Protocol = 17 $\wedge$ x.TCP.DstPort $\in$ {500}) $\vee$ (x.IP.Protocol $\in$ {50,51}) $\}$. □

2.2 Inferencing Algorithm

At a high-level, the inference algorithm works as follows: Initially, each network area with unknown classification is assigned all possible classifications. The inference algorithm successively excludes potential classifications for a network area if the actually allowed network flows between that network area and others would contradict the security policy for those classifications. The inferencing is based on the assumption that the actually allowed network flows are a subset of the flows permitted by policy. Extending our approach to work without this assumption is the focus of ongoing work.

We now describe the algorithm. Let $\mathcal{A}$ = {a_1, a_2, ... a_n} be the set of network areas. For a given invocation of the algorithm, the number of network areas is fixed. However, the composition of A can vary between successive invocations of the algorithm, i.e., stages of our approach. Let $\mathcal{C}$ = {c_1, c_2, ... c_n}, where c_i is the set of possible colors (i.e., classifications) for area a_i. The values of c_i are drawn from a domain $\mathcal{D}$ of colors, defined by the enterprise policy. For example, for the simplistic enterprise shown in Figure 1, $\mathcal{D}$ = {Blue, Yellow, Red}, with the elements representing intranet, extranet, and opennet respectively. If a_i's color is known a priori, then c_i is initialized with that value. Otherwise, c_i is initialized with all values in $\mathcal{D}$ (for the first stage), or with values left after the previous stage (for all stages other than the first stage). The set $\mathcal{A}$ must include at least two elements that belong to different colors, and whose colors are known in advance. Those elements serve as the baseline for comparison. This requirement can be easily satisfied in practice by considering one known subnet on the Internet and another on the Intranet.

Let $\mathcal{N}(a_i, a_j)$ be the feasibility set of actually allowed network flows from area a_i to a_j as indicated by data from various collection methods. Let $\mathcal{P}(c_i, c_j)$ be the feasibility set of packets from an area of color c_i to another of color c_j that are permitted by the security policy.

During each iteration, a color α is allowed to exist as a possible color for network area a_i if there is at least one color $\beta \in c_j$ such that the feasibility set of actually allowed packets between areas a_i and a_j is a subset of the feasibility set for the colors α and β. Otherwise, α

Algorithm 1: Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$

```
1.1  repeat
1.2      color_eliminated ← false
         {indicates whether any color was eliminated in this iteration}
1.3      foreach network area a_i ∈ A that has |c_i| > 1 do
1.4          foreach color α ∈ c_i do
1.5              foreach network area a_j ∈ A, where i ≠ j, do
1.6                  α_possible ← false
                     {indicates whether α is a possible color for a_i}
1.7                  foreach color β in c_j do
1.8                      if N(a_i, a_j) ⊆
                         P(α, β) ∧ N(a_j, a_i) ⊆ P(β, α) then
1.9                          α_possible ← true
1.10                 if α_possible = false then
1.11                     c_i ← c_i − α
1.12                     color_eliminated ← true
     until color_eliminated = false
```

is eliminated from the list of possible colors for a_i. The algorithm iterates until no color elimination is possible.

Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$ may be invoked multiple times during the course of the incremental discovery process. Specifically, it may be invoked at most once during the analysis phase of each stage; at most once, because other types of analysis not involving the algorithm may be performed. For example, hosts belonging to the same subnet may be identified and aggregated into one network area. Similarly, two network areas that are observed to have unrestrained connectivity without any intermediate enterprise firewalls may be merged into one network area. Similarly, new network areas of interest may be revealed by analyzing new connection logs. Such analysis may alter the sets $\mathcal{A}$ and $\mathcal{C}$ between successive invocations of Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$.

If two network areas a_i and a_j are being merged, then the set $\mathcal{A}$ will be changed to $\mathcal{A} = \mathcal{A} \setminus \{a_i, a_j\} \cup \{a_k\}$. Also, set $\mathcal{C}$ will be changed to $\mathcal{C} = \mathcal{C} \setminus \{c_i, c_j\} \cup \{c_k\}$, where the color of the merged network area a_k is given by $c_k = c_i \cap c_j$. The feasibility sets for a_k with respect to another network area a_l are given by

$$\mathcal{F}_{a_k->a_l} = \mathcal{F}_{a_i->a_l} \cup \mathcal{F}_{a_j->a_l} = \{x | f_{il}(x) \vee f_{jl}(x)\}$$
$$\mathcal{F}_{a_l->a_k} = \mathcal{F}_{a_l->a_i} \cup \mathcal{F}_{a_l->a_j} = \{x | f_{li}(x) \vee f_{lj}(x)\}$$

2.3 CSP-Based Solution

The comparison in line 1.8 of Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$ considers only two network areas at a time. As shown in Example 3 below, this may result in certain possible color eliminations being overlooked by Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$. To address this issue, we formulate a finite-domain CSP [7] that can be then solved using a general-purpose CSP solver.

Variables: $\mathcal{C} = \{c_1, c_2, \ldots c_n\}$

Domain of Values: $\mathcal{D}$

Constraints: $\mathcal{N}(a_i, a_j) \subseteq \mathcal{P}(c_i, c_j)$, where $1 \leq i, k \leq n$

A *complete assignment* to a CSP is defined as one in which every variable is mentioned, and a *solution* to a

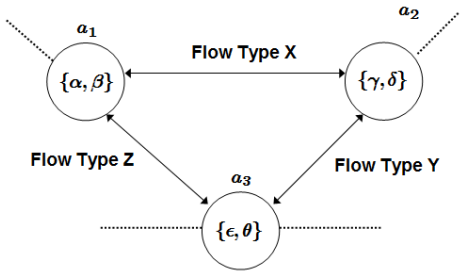


Figure 2: Output of Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$

Src \ Dst	α	β	γ	δ	ϵ	θ
α	**true**	**false**	Flow X	Flow X	Flow Z	**false**
β	**false**	**true**	Flow X	**false**	**false**	Flow Z
γ	Flow X	Flow X	**true**	**false**	Flow Y	**false**
δ	Flow X	**false**	**false**	**true**	**false**	Flow Y
ϵ	Flow Z	**false**	Flow Y	**false**	**true**	**false**
θ	**false**	Flow Z	**false**	Flow Y	**false**	**true**

Table 1: Predicates for Feasibility Sets Representing Security Policy (i.e., $\mathcal{P}$) for Example 3. Predicate "Flow X" holds for all packets that are of flow type X.

CSP is a complete assignment that satisfies all the constraints [7]. For the above CSP, each variable c_i has a domain $\mathcal{D}$ of possible values, and the number of possible complete assignments is $\prod_{i=1}^{n} c_i = \mathcal{O}(d^n)$, which is exponential in the number of variables. Here, $d = |\mathcal{D}|$. The worst-case complexity of Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$ can be shown to be a much smaller $\mathcal{O}(n^3 d^3 k)$. Here, k is the maximum number of clauses in the predicate defining $\mathcal{N}(a_i, a_j)$ or $\mathcal{P}(c_i, c_j)$ for any $1 \leq i, j \leq j$. Thus, it is typically much more efficient to first run Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$ and use the output of the algorithm to narrow down the domain of possible values for each c_i. Essentially, each color elimination done in advance using Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$ adds a unary constraint to the CSP that restricts the value of some c_i.

Example 3. Consider the situation shown in Figure 2 af-

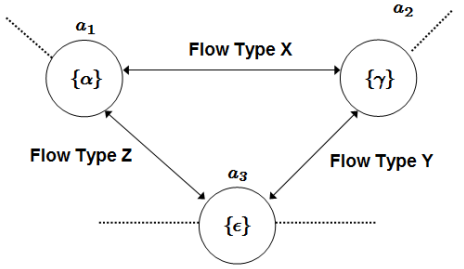


Figure 3: Further Color Elimination using CSP Solver

ter running Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$. Here, the domain $\mathcal{C}$ of colors is $\{\alpha, \beta, \gamma, \delta, \epsilon, \theta\}$ (i.e., d = 6) and a subsection of the network topology is shown highlighting three areas a_1, a_2, and a_3 among the n network areas. The edges are labeled with the types of network flows observed between each pair of nodes. For instance, Flow Type X was observed between a_1 and a_2. Table 1 shows the enterprise-level flow control policy between various classifications expressed as feasibility sets. Per Table 1, this implies that $c_1 = \{\alpha, \beta\}$ and $c_2 = \{\gamma, \delta\}$. Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$ will not be able to eliminate further colors beyond what is shown in Figure 2. On the other hand, if the output of the algorithm were used to initialize the domain values $\{c_i | 1 \leq i \leq n\}$ and then fed to a CSP solver, then the solver would consider the following six constraints simultaneously:

$\{ x \mid x \in \text{Flow X}\} \subseteq \mathcal{P}(c_1, c_2)$
$\{ x \mid x \in \text{Flow X}\} \subseteq \mathcal{P}(c_2, c_1)$
$\{ x \mid x \in \text{Flow Y}\} \subseteq \mathcal{P}(c_2, c_3)$
$\{ x \mid x \in \text{Flow Y}\} \subseteq \mathcal{P}(c_3, c_2)$
$\{ x \mid x \in \text{Flow Z}\} \subseteq \mathcal{P}(c_3, c_1)$
$\{ x \mid x \in \text{Flow Z}\} \subseteq \mathcal{P}(c_1, c_3)$

As a result, the solver would arrive at the solution $\{c_1 = \alpha, c_2 = \gamma, c_3 = \epsilon\}$ (Figure 3). It is important to note that if only the CSP solver were used (i.e., without first running Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$, then a total of $6^3 = 216$ possible complete assignments would have been considered for the three network areas. However, when the output of the algorithm is used to initialize the domain values, the number of possible color assignments for the three network areas reduces to just 8.

3 Implementation and Case Study

We have implemented a prototype of the solution approach described above as a tool called *BlueGates*. The tool is capable of analyzing actually allowed network flows from individual hosts, connectivity probes between peer devices, and CISCO PIX/ASA [4] firewall configuration files. Based on the analysis results, the tool derives feasibility sets to represent configuration information, and compares them with feasibility sets derived from an XML representation of the enterprise policy.

We now use a case study to illustrate our approach. The case study represents an abstracted subset of a real-life, operational enterprise environment. Table 2 shows the enterprise-level flow policy, i.e., the predicates for feasibility sets with the domain of colors, $\mathcal{D}$ = {Blue, Green, Yellow, Red} (indicated by the letters B, G, Y, and R respectively in Figure 4). In the particular enterprise environment we considered, these colors roughly corresponded to the classifications intranet, secure extranet, extranet, and Internet, respectively. Predicate "Flow X" holds for all packets with characteristics of flow type X. Thus, "Flow Standard Auth" holds for all packets that are part of a standard authentication flow (such as the FTP or Telnet protocol). "Flow Strong Auth" holds for all packets that are part of a strong authentication flow (such as the HTTPS or SSH protocol).

Src \ Dst	$Blue$	$Green$	$Yellow$	Red
$Blue$	**true**	**true**	**true**	**true**
$Green$	Flow **Standard Auth**	**true**	**true**	**true**
$Yellow$	Flow **Strong Auth**	Flow **Strong Auth**	**true**	**true**
Red	**false**	**false**	**true**	**true**

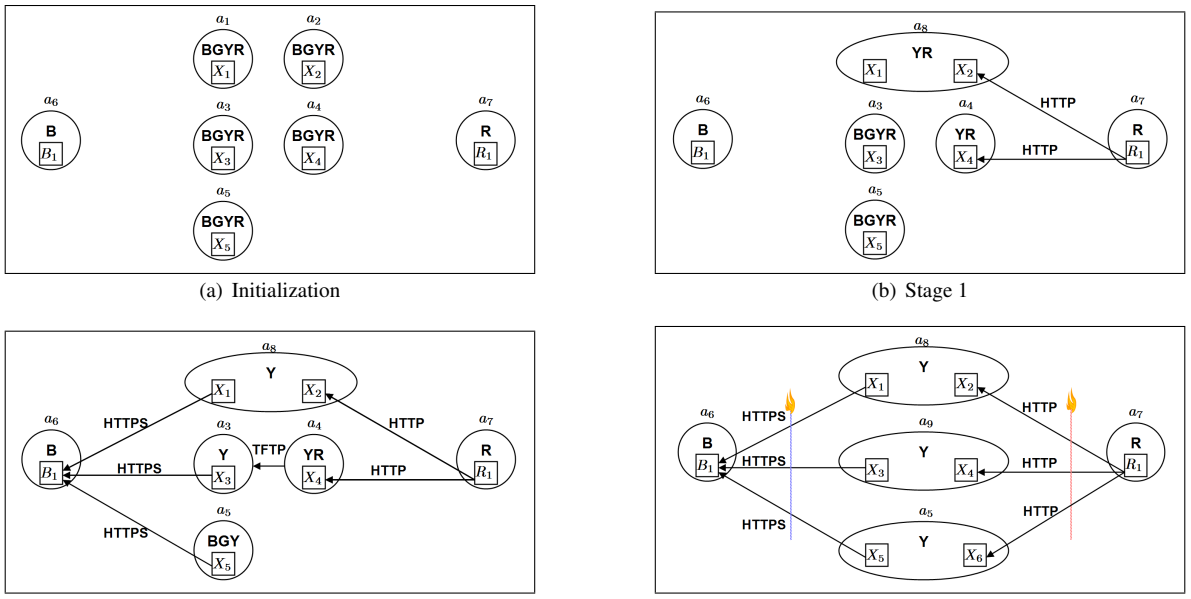
Table 2: Predicates for Feasibility Sets Representing Security Policy (i.e., $\mathcal{P}$).

Initialization: Figure 4(a) shows the stage before any discovery in which we are given seven hosts X_1, X_2, X_3,, X_4, X_5, B_1, and R_1. The colors for the hosts X_i are initially unknown, whereas B_1 is of color Blue and R_1 is of color Red. In the absence of any other information, each network area is initialized to contain a single host. Thus, the set $\mathcal{A}$ of network areas is initialized to $\{a_1,\ldots,a_7\}$ and $\mathcal{C}$ is initialized to $\{c_1,\ldots,c_7\}$. Here, network area a_i = {Host X_i}, for $1 \leq i \leq 5$, whose color c_i is unknown and therefore initialized to the set $\mathcal{D}$. Further, a_6 = {Host B_1} whose color c_6 = {Blue}, and a_7 = {Host R_1} whose color c_7 = {Red}.

Stage 1: Figure 4(b) shows the first stage of discovery, which involves analysis of `netstat` files obtained from the individual hosts. Analysis of these files by the BlueGates tool has established that hosts X_1 and X_2 belong to the same subnet. Consequently, a_1 and a_2 are merged into a single area a_8, whose color $c_8 = c_1 \cap c_2$. Thus, $\mathcal{A}$ = $\{a_3,\ldots,a_8\}$ and $\mathcal{C}$ becomes $\{c_3,\ldots,c_8\}$. The analysis also reveals the existence of active (unauthenticated) HTTP connections from host R_1 to X_2 and X_4; the feasibility sets $\mathcal{N}$ are updated accordingly before invoking Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$. When the algorithm completes execution, c_4 and c_8 are reduced to the set {Yellow, Red}.

Stage 2: Figure 4(c) shows the second stage of discovery, which involves active probing and analyzing connectivity between hosts. BlueGates' analysis has established that (i) hosts X_1, X_3, and X_5 can communicate with host B_1 using the HTTPS protocol (i.e., strong authentication), and (ii) TFTP (a basic file transfer protocol with no user authentication) is allowed from host X_4 to X_3. The feasibility sets $\mathcal{N}$ are updated accordingly before invoking Algorithm CLASSIFY $(\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N})$. When the algorithm completes execution, c_3 and c_8 are reduced to the set {Yellow}.

Stage 3: Figure 4(d) shows the third stage of discovery, in which intermediate enterprise firewalls are identified and their configuration files analyzed by BlueGates. The analysis reveals that there is no firewall separating the traffic between X_3 and X_4. Consequently, network



(a) Initialization

(b) Stage 1

(c) Stage 2

(d) Stage 3

Figure 4: Incremental Discovery

areas a_3 and a_4 are merged into a single area a_9, whose color c_9 = {Yellow}. Thus, $\mathcal{A}$ = {a_5,. . .,a_9} and $\mathcal{C}$ becomes {c_5,. . .,c_9}. Further, the firewall configuration analysis reveals that HTTP traffic is allowed from R_1 to a previously unknown host X_6, which is in the same subnet as host X_5. Consequently, network area a_5 is expanded to the set {Host X_5, Host X_6}. The feasibility sets $\mathcal{N}$ is updated accordingly before invoking Algorithm CLASSIFY ($\mathcal{C}, \mathcal{A}, \mathcal{P}, \mathcal{N}$). When the algorithm completes execution, c_5 is reduced to the set {Yellow}.

4 Conclusion

We described a systematic and semi-automated approach for discovering security zone classifications of devices in an enterprise environment. We use a common format (based on feasibility sets [5]) to represent both network flows actually allowed (as seen in configuration settings or traffic observed) and the flows permitted by the enterprise security policy. As part of our approach, we described an elimination-based inference algorithm and an alternative based on the constraint satisfaction problem. For improved efficiency and reduced interference to normal network operation, we described a staged approach to collecting information about actually allowed flows.

One limitation of our approach is the assumption that the actually allowed network flows are a subset of the network flows allowed by enterprise security policy. The validity of this assumption can be increased by applying our approach to obtain the classifications for network areas in a small subset of (rather than) the entire enterprise network, and then repeating this process for different subsets of the enterprise network. When this assumption does not hold, our approach will indicate that there is a non-compliance (e.g., the CSP solver will indicate that no solution is possible). However, it will not pinpoint the exact sources of the non-compliance. Our ongoing work focuses on addressing this limitation. One promising approach that we are exploring involves incrementally adding network areas to the input set $\mathcal{A}$ until the inference algorithm or the CSP solver indicates a non-compliance situation. Future work will also include applying this approach in large-scale IT environments and evaluating its performance and effectiveness.

References

[1] AL-SHAER, E., AND HAMED, H. Modeling and Management of Firewall Policies. *IEEE Transactions on Network and Service Management 1* (April 2004), 2–10.

[2] AL-SHAER, E., AND HAMED, H. Taxonomy of Conflicts in Network Security Policies. *IEEE Communications 44* (March 2006), 134–141.

[3] CARACAS, A., DECHOUNIOTIS, D., FUSSENEGGER, S., GANTENBEIN, D., AND KIND, A. Mining Semantic Relations using NetFlow. In *3rd IEEE/IFIP Intl. Workshop on Business-Driven IT Management (BDIM)* (2008), pp. 110–111.

[4] CISCO SYSTEMS. Cisco PIX/ASA Products. `http://www.cisco.com/en/US/products/ps6120`, 2010.

[5] GUTTMAN, J. D., AND HERZOG, A. L. Rigorous automated network security management. *International Journal for Information Security 4*, 1-2 (2005), 29–48.

[6] LYON, G. F. Nmap Network Scanning. `http://nmap.org/book/toc.html`, 2010.

[7] TSANG, E. *Foundations of Constraint Satisfaction*, 2nd ed. Academic Press, 1993.

[8] TUFIN TECHNOLOGIES. SecureTrack Firewall Operations Management. `http://www.tufin.com`, 2010.

[9] WIRESHARK. The Wireshark Tool. `http://www.wireshark.org/`, 2010.