Logic-based methodology to help security architects in eliciting high-level network security requirements

Romain Laborde
University Paul Sabatier
Toulouse, France
romain.laborde@irit.fr

Sravani Teja Bulusu
University Paul Sabatier
Toulouse, France
sbulusu@irit.fr

Ahmad Samer Wazan
University Paul Sabatier
Toulouse, France
ahmad-samer.wazan@irit.fr

François Barrère
University Paul Sabatier
Toulouse, France
francois.barrere@irit.fr

Abdelmalek Benzekri
University Paul Sabatier
Toulouse, France
abdelmalek.benzekri@irit.fr

ABSTRACT

In this paper[1], we propose a security methodology that automates the process of security zone specification and high-level network security requirements elicitation. We define a set of formalized rules derived from the principles of complete mediation, least privileges and the Clark-Wilson lite formal model making our approach traceable and verifiable. We implemented the methodology in Answer Set Programming to automatically compute an optimal network security zone model considering the cost of the security solution. A use case study of an e-commerce enterprise network infrastructure illustrates our methodology.

CCS CONCEPTS

• **Security and privacy~Security requirements** • Security and privacy~Network security

KEYWORDS

Network Security Requirements, Security Zoning, Integrity Model, Answer Set Programming

ACM Reference format:

R. Laborde, S.T. Bulusu, A.S. Wazan, F. Barrère, A. Benzekri. 2019. In *Proceedings of ACM SAC Conference, Limassol, Cyprus, April 8-12, 2019 (SAC'19)*, 10 pages. https://doi.org/10.1145/3297280.3297437

SAC'19, April 8-12, 2019, Limassol, Cyprus

https://doi.org/10.1145/3297280.3297437

1 INTRODUCTION

Network security is a broad topic to address at multiple layers. Indeed, securing a network covers controlling physical access to allow only authorized devices to communicate through the network, controlling and protecting data flows and protecting end systems from being compromised [19,20]. Nevertheless, strengthening the security increases also the price of the network accordingly. Thereby, a compromise between security and price should be established and organizations/industries are seeking cost-effective security solutions that protect networks against malicious attacks while meeting the business requirements.

In this context, network security requirement engineering is a key activity since bad network security requirements can lead to ineffective and costly security or worth security holes in network security design [1]. However, the requirement engineering research community has neglected network security in spite of its vital importance. As a result, current security requirement engineering methodologies don't provide good support to derive network security requirements [6–8]. Thus, a methodology suitable to network security is mandatory.

The current practice for eliciting and analyzing early network security requirements is driven by network zoning [10]. Network zoning (a.k.a. network segmentation) is a key defense-in-depth strategy that segregates and protects key company assets and limits lateral movements of attackers across corporate network in case of intrusion. Partitioning networks is also effective in reducing the scope of audits for regulations. Security zones represent different trust levels, which exhibit the criticality of the systems within the zones. Each security zone constitutes a logical grouping of security entities that are subject to similar protection requirements (e.g., data confidentiality and integrity, access control, audit, etc.). Determining security zones and respective trust levels is a preliminary step for security architects to derive other network security requirements such physical access, dataflow control and protection, and end-systems protection [27,28].

In the literature, several research works and best practice approaches propose various zone classification schemes and patterns [10,21,27]. Nevertheless, this process takes place in an ad-hoc manner and do not integrate a rigorous approach to specify security zones and rules governing zones interactions. Being manual, the design of security zones depends only on the expertise of the architects who may forget some details while specifying the zone model, which will impact the quality of derived network security requirements [5]. As part of the IREHDO2 research project, we conducted interviews of senior security requirements engineers that revealed network security architects are looking for a methodological approach to, at least, consolidate the network security architectures they propose. Indeed, the task of verifying and validating the network security requirements with regards to business security requirements is tedious and challenging [12]. How to ensure that the proposed network zoning is correct and cost-effective? How to ensure that no network security requirement is missing or irrelevant?

In this paper, we describe a methodology that automates parts of the process of eliciting security zones and derived network security requirements by using a set of formalized rules derived from three well-established security principles (complete mediation, least privileges and the Clark-Wilson lite formal security model), thereby leaving less space to human errors. We implemented the rules in Answer Set Programming (ASP [14]) and provide a tool that automates the calculation cost-optimal security zone models. We illustrate our methodology using an example case study of e-commerce enterprise network infrastructure.

The rest of the article is structured as follows. In section 2, we describe the example case study. Section 3 presents the concepts related to our approach. Section 4 presents the steps in our zone modelling methodology. Section 5 illustrates our methodology by applying it to an e-commerce scenario. Section 6 deals with related works. Finally, we conclude our work and propose future research directions in section 7.

2 E-COMMERCE ENTERPRISE NETWORK CASE STUDY

We consider an e-commerce enterprise network case study [2] as a running example **(**Figure 1**).** In brief, the initial network architecture (see Figure 1(a)) consists of web server, DNS server, application server, database server, and accountability server. The employees are distinguished as administrators and standard users, who can connect to the network through LAN or WIFI. If the employees are outside the enterprise, they can remotely connect to the enterprise network. Finally, when the clients visit the enterprise, they are allowed to connect to the web through a dedicated WIFI network.

The accountability server is highly critical as it manages the financial information of the company (e.g., salaries of employers). The web server hosts the e-commerce web site. Therefore, it is also critical because a deny of service attack will highly impact the business of the company. Finally, the web server requires interactions with the application and database servers to provide the e-commerce service. Consequently, they are also highly important assets for the business especially the database for which data integrity is primordial.

The network security requirements engineers of the company propose the security architecture in Figure 1(b) (and described in [2]). The solution reflects some best practice guidelines by defining some zones such as DMZ zone, user's zones, etc. For instance, the accountability server is isolated in a separate zone as it is highly critical. However, it is not clear on how the architects concluded to this solution? This approach requires additional arguments to demonstrate the solution meets the elicited risks. In addition, this diagram does not deal with the cost-effectiveness ratio of this solution. We believe that a formal approach justifying the transition from the problem to the solution is required for a traceable and verifiable security zone specification process.

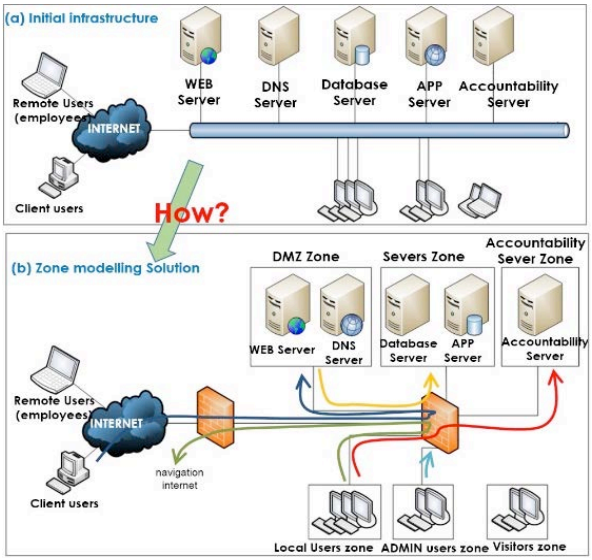

Figure 1. E-commerce example case study [2]

3 PROPOSED METHODOLOGY CONCEPTS

In this section, we present the main concepts to develop our methodology.

3.1 Analysing the risk for the enterprise

We mainly consider three main elements: security domains, agents and security zones. A *security domain* represents the organizational authority which controls and manages the entities (i.e., servers, software, data, users, etc.) that belong to it. We call these entities as *agents*. Furthermore, a security domain can be refined into sub-domains highlighting different policies or situations within the same organization. In our scenario, there are two domains: the enterprise domain and the rest of the world named Internet. The enterprise domain is itself divided into two sub-domains (see Figure 2): the internal sub-domain that consists in the assets within enterprise premises and the external sub-

domain containing remote employees working outside the premises of the enterprise.

We assume the functional requirements have been elicited using an agent-based approach (such as User Requirements Notation [18] or Socio-Technical Security Modeling Language [11]). Agents are concrete entities (represented by circles with a straight line) that have or are assigned to objectives (depicted by squares within circles attached to agents). Agents with same characteristics can be abstracted with the notion of *role* (noted by a circle with a circle with a curve, e.g., remote users). However, we have augmented the notation to differentiate two categories of agents/roles. *System agents* (represented by pink circles) refer to entities under direct control such as software/hardware systems that are developed and/or maintained by the enterprise. *Environment agents* (represented by violet circles) are not under direct control and refer to humans or some off-the-shelf software.

Finally, *security zones* constitute logical grouping of agents with common protection requirements. As a consequence, our methodology mainly aims at grouping *agents* within *security zones* managed in *security domains* and eliciting the related security requirements.

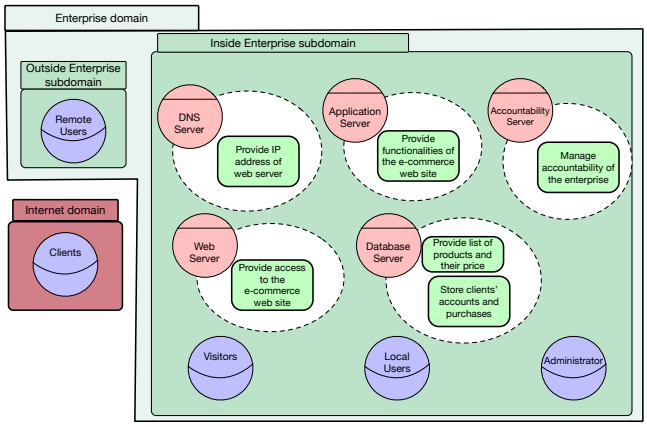

Figure 2. Sample of functional requirements

We consider that some external risk analysis has assessed security domains and agents to provide domain control capability, trust of environment agents and criticality of system agents. *Security domain control capability* describes the maturity of an enterprise to deploy security controls and/or its capability to control a given environment. In our scenario, the enterprise *inside sub-domain* refers to the office premises of the e-commerce organization that is physically secured. We consider also that the enterprise is mature with well-trained employees. Thus, this sub domain is *well controlled*. The *outside sub domain* consists in employees working from their home using laptops provided by the office. Then, it is *less controlled*. The *Internet domain* being outside the control of the enterprise is *uncontrolled*.

Environment agents are given a *trust* level, which specifies the degree of the trustworthiness over the expected behavior of environment agents in a given context. For instance, in our use case, the administrator is highly trusted because this person is well-known and qualified. Local employees are only trusted because this role refers to more people. Visitors are partially trusted because they are known people and the reception staff verifies the visitor's identity card which must be surrendered in exchange for a wearable badge.

Finally, system agents are evaluated based on their *criticality* levels. Criticality level determines the sensitivity to threats and the risk impact of system agents' goals on the overall business. For instance, '*providing access to the e-commerce web site*' is critical for the business of the enterprise. Thus, the *web server*, which is assigned to this goal, is *critical* too. The DNS and the application servers have the same criticality as the web server because the risk is the same if one of them cannot achieve its associated goal. The *database server* is *highly critical*. On one side, its goal '*provide list of products and their price*' is only critical. On the other side, the goal '*store clients' accounts and purchase*' is highly critical because a threat on this goal will have a strong impact on the reputation of the enterprise. Finally, the *accountability server* is *vital* because it can lead to bankruptcy if the enterprise can't manage its accountability.

3.2 The three core security principles

We construct our methodology based on three well established principles. First of all, the principle of *complete mediation* [26] stipulates controlling every accesses. Applying to the context of network security zones, it means that 'every data flows between zones must be controlled by a security mechanism'. The principle of *least privileges* [26] requires to limit users to access only what is necessary for their legitimate purpose. We translate this principle in the context of network security zoning as 'a user can access a zone only if he is granting access to all the services within the zone'. Here we map the services level privileges to the network level privileges. This limit the propagation of an attack on a service to only the zone.

Finally, formal models of *integrity* foster to avoid critical systems to consume untrusted/fake information. We employ also the concept of integrity to unify capability, trust and criticality to facilitate the integration of risk analysis concepts to our network requirement analysis context.

First, integrity of an agent reflects the assurance of an expected behavior. This fits with the concept of trust related to environment agents. The more trust the enterprise can have on an entity, the more it expects a given agent's behavior, and hence, the more integrity it believes the agent has. As consequence, the trust assessment can be transformed into a maximum integrity value representing an assumption. Similarly, system agents' criticality can be expressed as a required integrity. Critical goals are required to be achieved; hence, it is a required behavior of the assigned system agent. It means criticality corresponds to the minimum integrity value required for a system agent. For instance, the accountability server being vital requires a high level of integrity while the web server, which is considered critical only for business, requires less integrity. Finally, the integrity of a

security domain correlates with the maximum integrity an enterprise can achieve with its control capability.

As a consequence, we assume the existence of some utility functions (Figure 3) that maps the control capability labels of domains, criticality and trust levels of agents into a unified scale of integrity values. Figure 3 shows an example assumed for rest of the paper.

Unifying risk and trust within the concept of integrity allows us also to integrate formal integrity security models with security zone modelling design principles to address the risks pertaining to traffic flows and information assurance. There exists several models of integrity such as Biba [3], Clark-Wilson [9], which propose abstract solutions to preserve the integrity of information flows. These models are widely used in current operating systems for improving the integrity protection of the information flows in inter-process communications (e.g., Microsoft Windows Integrity Mechanism [22]). We have chosen to adapt the model of Clark-Wilson lite [29] (lighter version of Clark-Wilson model) for verifying the integrity property of traffic flows traversing multiple zones because this model explicitly considers information flows and then is more suitable to network data flows analysis.

Domain	**Control capability**	**Integrity**
Enterprise internal sub-domain	Highly restricted	5
Enterprise external sub-domain	Less controlled	2
Internet	Uncontrolled	1
System agents	**Criticality**	**Integrity**
WEB server	Critical	3
DNS server	Critical	3
APP sever	Critical	3
Database server	Highly Critical	4
Accountability server	Vital	5
Environment agents	**Trust**	**Integrity**
Admin user	Highly trusted	4
Local users	Trusted	3
Visitors	Partially trusted	2
Remote users	Partially trusted	2
Client users	Not trusted	1

Figure 3. Integrity values of domains and agents for the example case study

The main idea of the integrity models is as follows. Entities take decisions based on input information (e.g. a program executes an algorithm based on its inputs). If input information is wrong, then the decision can be wrong too. Therefore, critical systems must read information with high levels of integrity only (i.e. high level of assurance) while non critical systems are not subject to such constraint. In CW-lite model, all information flowing from low integrity subjects to high integrity subjects must be filtered. Here, the integrity filters correspond to integrity validation procedures that sanitize information or block it. For instance, in network security analysis context, an integrity validation filter can be a web application firewall that checks SQL statements or URL formats. Integrity models complete the principle of complete mediation by checking the content of flows.

CW-lite model places the integrity validation filters at the receiving subject's side and express the information flow control as follows (Figure 4): "if a subject s receives an information flow from a subject s_i at interface I, then either there is an integrity validation filter at interface I or the integrity level of s_i is greater or equal to the integrity of subject s".

$$flow(s_i, s, I) \wedge \neg filter(s, I) \rightarrow (int(s_i) \geq int(s))$$

Figure 4. CW-lite security filtering rule [29]

4 OUR PROPOSED METHODOLOGY

Our zone modelling methodology (see Figure 5) is divided into two main steps: (1) Determining the security zones and integrity validation filters and (2) Identifying data flows integrity requirements and data flows access control filters.

The initial input of the first step is the set of security domains, the set of agents, the integrity levels of domains and agents, and the data flows between agents. As a result of step1, our process computes the security zones and the integrity validation filters. Then, the security architect provides additional information about the media of communication (i.e., the networks) between zones and launches the second step. The final result is a set of network security requirements that are a set security zones, integrity validation filters, agents' integrity requirements, access control filters, and integrity data flow protection requirements. For implementation, we formalized step1 and step2 in Answer Set Programming (ASP) [14]. ASP is a declarative logic based approach that facilitates the solving of difficult search problems by computing answer sets through stable model semantics. We defined the rules at each step and the ASP solver determines the set solutions (called answers) that are compliant with the rules and the input. This makes our process traceable and verifiable.

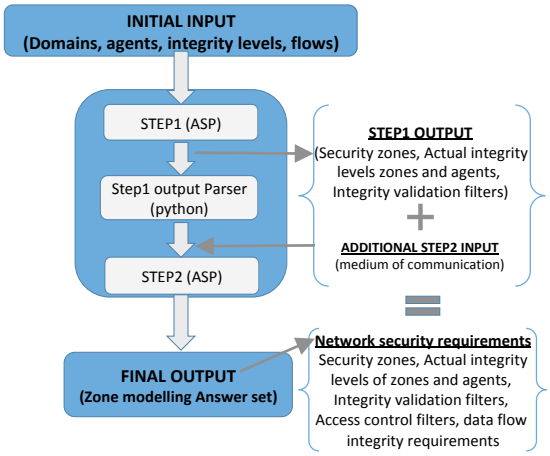

Figure 5. Our zone modelling methodology approach overview

In the following, we discuss in detail the modelling rules at step1 and step2. The implementation of our zone modelling solution will be discussed separately in section 5.

4.1 Step 1: Specifying zones and filtered flows

The main goal of this step is specifying zones and identifying integrity validation filters. The process starts with a system as a set of domains, zones and agents and we represent it as follows:
$S = <DOMAIN, ZONE, AGENT, FLOW, INSIDE^{D}_{Z}, INSIDE^{D}_{A}, INSIDE^{Z}_{A}, Int, Int_{max}, Int_{min}, Int_{actual}, Agent_{Server}, Agent_{Client}>$
Where:

- DOMAIN is the set of security domains.
- ZONE is the set of security zones.
- AGENT is the set of agents, named after entities. AGENT = ENV_AGENT ∪ SYST_AGENT with ENV_AGENT and SYST_AGENT being the set of environment and system agents such that ENV_AGENT ∩ SYST_AGENT=∅.
- $Agent_{Server}$: AGENT → {TRUE, FALSE} states if an agent is a server (e.g., web server).
- $Agent_{Client}$: AGENT → {TRUE, FALSE} states if an agent is a client (e.g., browser).
- FLOW ⊆ AGENT × AGENT, is the set of allowed flows of information.
- $INSIDE^{D}_{z} \subseteq$ ZONE × DOMAIN is a relation that states a zone is inside a domain.
- $INSIDE^{D}_{A} \subseteq$ AGENT × DOMAIN is a relation that states an agent is inside a domain.
- $INSIDE^{Z}_{A} \subseteq$ AGENT × ZONE is a relation that states an agent is inside a zone.
- Int: DOMAIN → ℕ returns the integrity level of a security domain. This value is directly derived from the control capability value.
- Int_{max}: ZONE ∪ AGENT → ℕ returns the maximum integrity of a zone or an agent. For environment agents, this value is directly derived from their trust value.
- Int_{min}: AGENT → ℕ returns the minimum integrity level of an agent. For system agents, this value is directly derived from the criticality value.
- Int_{actual}: ZONE ∪ AGENT → ℕ returns the actual integrity of a zone or an agent, which are the final integrity values computed at the end of the process.
- integrity-validation-filter(a:AGENT,f:FLOW,val_1:ℕ,val_2:ℕ) states integrity validation requirements such that *integrity-validation-filter(a, f, val_1, val_2)* means some integrity protection mechanism at agent *a* must sanitize dataflow *f* with an integrity level of val_1 to achieve a data assurance level of val_2.

In other words, Int, Int_{max} and Int_{min} represent the integrity mapping functions (Figure 3). Accordingly, we define the rules of step1 as follows:

RULE 1: Every agent is inside a domain.
$\forall a \in AGENT, \exists d \in DOMAIN \mid (a,d) \in INSIDE^{D}_{A}$

RULE 2: Every security domain contains at least one security zone.
$\forall d \in DOMAIN, card(\{z \mid z \in ZONE, (z, d) \in INSIDE^{D}_{Z}\}) \geq 1$

RULE 3: The maximum integrity level of a security zone is equal to the integrity level of the domain. This is because, a domain controls zone and therefore we cannot have more assurance on a zone than the controlling domain.
$\forall d \in DOMAIN, \forall z \in ZONE, (d, z) \in INSIDE^{D}_{Z} \Rightarrow Int_{max}(z) = Int(d)$

RULE 4: Similar to Rule 3, the maximum integrity level of an agent is equal to the integrity level of domain.
$\forall d \in DOMAIN, \forall a \in AGENT, (a, d) \in INSIDE^{D}_{A} \Rightarrow Int_{actual}(a) \leq Int(d)$

RULE 5: The actual integrity of a zone cannot be greater than its maximum integrity.
$\forall z \in ZONE, Int_{actual}(z) \leq Int_{max}(z)$

RULE 6: The actual integrity of agents must be between the maximum and the minimum integrity levels of the agents.
$\forall a \in AGENT, Int_{min}(a) \leq Int_{actual}(a) \leq Int_{max}(a)$

RULE 7: The actual integrity levels of an agent are same as that of its residing zone.
$\forall a \in AGENT, \forall z \in ZONE, (a,z) \in INSIDE^{Z}_{A}, \Rightarrow Int_{actual}(a) = Int_{actual}(z)$

RULE 8 - CW-Lite: The actual integrity levels of the interacting agents must adhere to the CW-lite integrity rule. In this way, an agent cannot have access to lower integrity information.
$\forall a_1,a_2 \in AGENT, (a_1, a_2) \in FLOW \wedge \neg$integrity-validation-filter(a_2,flow(a_1,a_2),
$int_{actual}(a_1), int_{actual}(a_2)) \Rightarrow Int_{actual}(a_1) \geq Int_{actual}(a_2)$

RULE 9 – Principle of complete mediation: Server agents and client agents cannot reside in same zone. Because, as per the zone modelling design principles, intra-zone interactions are usually not analyzed. With reference the security design principle of complete mediation rule, every access to every object must be validated [26]. Therefore, if server and client reside in the same zone there will be a conflict.
$\forall c,s \in AGENT, \forall z_1,z_2 \in ZONE, (c, z_1) \in INSIDE^{Z}_{A}, (s, z_2) \in INSIDE^{Z}_{A}, Agent_{Server}(s), Agent_{Client}(c) \Rightarrow z_1 \neq z_2$

RULE 10 – Principle of least privileges: The least privileges principle aims at minimizing the permissions of users to the minimum required for accomplishing their tasks [26]. From a network perspective, a client agent can send flows in a security zone only if he can send flows to all the server agents within that zone. In this way, we map network level privileges to services level privileges. Indeed, server agents will be grouped in zones according to users' privileges.

$\forall c,s_1,s_2 \in$ AGENT, $Agent_{Client}(c)$, $Agent_{Server}(s_1)$, $Agent_{Server}(s2)$, $\forall z_1,z_2 \in$ ZONE, $(s_1, z_1) \in INSIDE^Z_A$, $(s_2, z_2) \in INSIDE^Z_A$, $(c,s_1) \in$ FLOW, $(c,s_2) \notin$ FLOW $\Rightarrow z_1 \neq z_2$

4.2 Step2: Specifying integrity requirements for the communication medium between zones

At the end of step1, the set of zones along with the integrity validation filters are determined. Step2 addresses the security issues related to inter-zone interactions, i.e., the protection of data flows through the network communication media (e.g., wired/wireless networks, etc.,) that connect the computed zones. The main goal of this step is to protect the integrity of data flows when traversing untrusted media of communication. Suitably, we complete our system model as follows:

S = <DOMAIN, ZONE, AGENT, FLOW, MEDIUM, $INSIDE^D_z$, $INSIDE^D_A$, $INSIDE^Z_A$, $INSIDE^D_M$, CONNECT, Int, Int_{max}, Int_{actual}>,

Where:

- MEDIUM is the set of media of communication.
- $INSIDE^D_M \subseteq$ MEDIUM × DOMAIN is a relation, which states that a medium of communication is in a domain.
- CONNECT $\subseteq$ MEDIUM × ZONE is a relation, which states that a zone is connected to a medium of communication.
- Int_{max}: ZONE ∪ AGENT ∪ MEDIUM → $\mathbb{N}$ returns the maximum integrity level of a security zone, agent or medium of communication.
- Int_{actual}: ZONE ∪ AGENT ∪ MEDIUM → $\mathbb{N}$ returns the actual integrity level of a security zone, agent or medium of communication.
- PATH $\subseteq$ FLOW × (ZONE ∪ MEDIUM) × (ZONE ∪ MEDIUM), is a relation that stores where flows are transiting with the constraint that (f, e_1, e_2) $\in$ PATH $\Rightarrow$ (e_1,e_2) $\in$ CONNECT $\vee$ (e_1,e_2) $\in$ CONNECT. For instance, *(f,m,z)* $\in$ *PATH* means that flow *f* transits between medium *m* to zone *z*.
- access-control-filter(c:CONNECT,f:FLOW) states access control requirements so that *access-control-filter(c,f)* means flow *f* must be permitted at connection point *c*.
- dataflow-integrity-protection(f: FLOW, e: ZONE ∪ MEDIUM, value: $\mathbb{N}$) states dataflow protection requirements such that *dataflow-integrity-protection(f,e,val)* means some protection mechanism must be applied on dataflow *f* over zone or medium *e* to preserve an integrity level of *val*.

Similar to domains, zones, and agent, a medium of communication *m* has two integrity levels: $Int_{min}(m)$, and $Int_{actual}(m)$. Accordingly, we add new rules to include constraints on media of communication:

RULE 11: Every zone must be connected to a medium of communication.

$\forall z \in$ ZONE, $\exists$ m $\in$ MEDIUM, (m, z) $\in$ CONNECT.

RULE 12: At each zone, there must be an access control filter that permits allowed flow of information. Not explicitly allowed flows are denied by default.

$\forall$ (f, e_1, e_2) $\in$ PATH, $e_1 \in$ MEDIUM $\wedge$ $e_2 \in$ ZONE $\Rightarrow$ access-control-filter((e_1,e_2), f)

And respectively:

$\forall$ (f, e_1, e_2) $\in$ PATH, $e_1 \in$ ZONE $\wedge$ $e_2 \in$ MEDIUM $\Rightarrow$ access-control-filter((e_2,e_1), f)

RULE 13: The actual integrity level of a medium of communication is equal to the minimum value between the integrity level of its domain, the initial trust of the medium (i.e., its maximum integrity), and the actual integrity levels of the connected zones.

$\forall$ m $\in$ MEDIUM, $Int_{actual}(m) = \min(\{Int(d) \mid d \in DOMAIN, (m,d) \in INSIDE^D_M\} \cup \{Int_{max}(m)\} \cup \{Int_{actual}(z) \mid z \in ZONE, (m,z) \in CONNECT\})$

RULE 14: A flow that transits over a medium or a zone, requires an integrity protection, if the integrity level of the medium or the zone is lower than the level of integrity of the flow.

$\forall (a_1,a_2) \in$ FLOW, $\forall$ $e_1,e_2 \in$ ZONE ∪ MEDIUM | (flow(a_1,a_2), e_1, e_2) $\in$ PATH, $\min(int_{actual}(a_1), int_{actual}(a_2)) > int_{actual}(e_1) \Rightarrow$ data-flow-integrity-protection(flow(a_1,a_2),e_1,$\min(int_{actual}(a_1), int_{actual}(a_2))$).

Respectively:

$\forall (a_1,a_2) \in$ FLOW, $\forall$ $e_1,e_2 \in$ ZONE ∪ MEDIUM | (flow(a_1,a_2), e_1, e_2) $\in$ PATH, $\min(int_{actual}(a_1), int_{actual}(a_2)) > int_{actual}(e_2) \Rightarrow$ data-flow-integrity-protection(flow(a_1,a_2),e_2,$\min(int_{actual}(a_1), int_{actual}(a_2))$).

5 IMPLEMENTATION

We implemented our methodology in ASP using the Clingo solver [15] and Python. In this section, we describe the security zone calculation of the scenario using our methodology by detailing each step of the process.

5.1 Step1 – Eliciting security zones and integrity validation filter requirements

5.1.1 Step1 input

At step1, the initial knowledge on the system concerning the agents, domains and their integrities needs to be provided as input. Figure 6 shows the graphical view.

In the example case study, we have considered a total of two domains. The enterprise domain is under the control of the e-commerce enterprise, while the Internet domain corresponds to the public domain, which is uncontrolled. The enterprise domain has two sub-domains. The inside enterprise sub-domain may refer to the office premises of the e-commerce organization that is physically secured. The outside sub domain consists in employees working from their home using laptops provided by the office. By applying the utility function described in Figure 3, these control capabilities are transformed into integrity values.

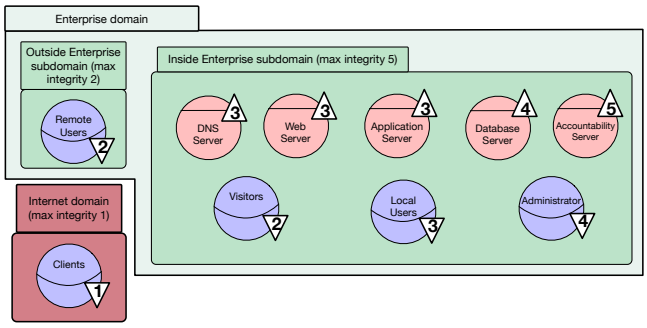

Figure 6. Step 1 – input

The integrity levels of system agents (depicted in triangular shapes) correspond to the minimum level of integrity that should be maintained in terms of security assurance. Likewise, the integrity levels of environment agents (depicted in inverted triangular shapes) correspond to the maximum level of integrity, which is expected as guaranteed. These integrity values are also determined based on the utility function in Figure 3. Finally, the initial input includes the list of permitted data flows between agents regarding the business objectives. Table 1 lists a sample of the data flows considered in the example case study. E.g., role *adminUser* must be able to send data flows to agent *accountabilityServer*.

Table 1. Step 1 input – sample of permitted data flows

```
Flow(adminUser,accountabilityServer)
Flow(localUsers,accountabilityServer)
Flow(remoteUsers,accountabilityServer)
Flow(accountabilityServer,adminUser)
Flow(accountabilityServer,localUsers)
Flow(accountabilityServer,remoteUsers)
```

5.1.2 Step1 output

Our tool automatically computes the security zones and the integrity validation filters according to the rules listed in section 4.1 (see Figure 7). The Clingo ASP solver can produce many solutions (i.e. zone models), which can equally satisfy all the rules. However, their costs of implementation can vary. From a broad view, the implementation cost is the summation of the cost of implementing the network security requirements to preserve the actual integrity levels of each zones and of upholding the integrity verification filtering requirements.

The tool determines the optimized solution using the ASP optimization statement *minimize*, to find the solution with minimal cost. Figure 8 depicts an example of cost formula where the Clingo solver searches for an optimal answer set that minimizes the sum of all values of entities' integrity levels (i.e., sum of actual integrity values of agents and integrity to be guaranteed by integrity validation filters). The cost calculus can vary for different organizations based on their logistics study similar to the varying cost effects of the Design Assurance Levels [24]. Thus, this cost formula is just an example and another cost computation formula more suitable to a specific organization can be specified.

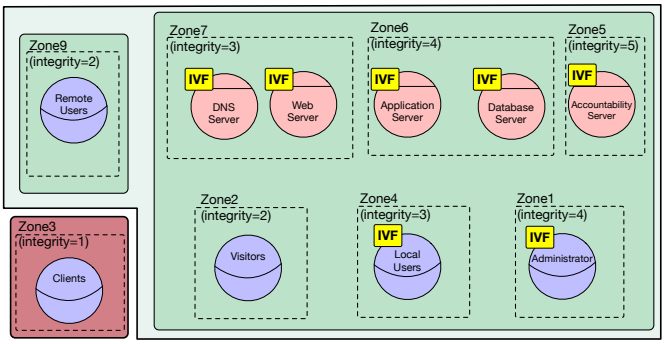

Figure 7. Step 1 – output

```
14 % Optimisation
15 #minimize {L,A:integrityActual(A,L);DSTL,filter(A,F,SRCL,DSTL):filter(A,F,SRCL,DSTL)}.
```

Figure 8. Example of a cost-based optimization rule

We executed our tool 30 times on a Mac Book Pro (2.8 GHz Core i7, 16GB RAM) and configured the solver to use 4 threads in parallel. The execution times to calculate the optimal solution based on our cost optimization rule varied between 5.8 seconds and 7.6 seconds.

Our zone model is abstract and design independent therefore does not restrict the technical solutions. It specifies a total of 8 zones with 6 inside the enterprise network, and 2 zones outside. It is interesting to note that our tool obtained the same zones as the proposed case study (Figure 1(b)).

Table 2. Step 1 output – Sample of IVF requirements

```
IVF(accountabilityServer,
      flow(localUsers,accountabilityServer),
      sanitize(3,5))
IVF(accountabilityServer,
     flow(remoteUsers,accountabilityServer),
     sanitize(2,5))
IVF(accountabilityServer,
      flow(adminUser, accountabilityServer),
      sanitize(4,5))
```

Our tool cannot classify the zone types like DMZ, restricted, etc. However, integrity values of zones provide zone integrity requirements. The resulting calculated integrity levels attached to zones also apply to all the agents within the zones. These integrity levels have to be interpreted as the pre-requisite requirements to achieve at minimum by the future security implementation related to software development processes, code practices, software verification and validation, etc. Indeed, in practice, there already exist code assurance level standards such as the DALs for aircraft systems requirements [4]. The higher the DAL is, the higher the assurance activities and design verification methods are

demanded. In our methodology context, the actual integrity levels of system agents exhibit similar characteristics as DALs.

Additionally, the tool identifies integrity validation filters (IVF) attached to agents. They are depicted by yellow squares in Figure 7. A sample from the set of IVF rules automatically generated by our tool is given in Table 2. For instance, *IVF(accountabilityServer,flow(localUsers,accountabilityServer), sanitize(3,5))* is a requirement stating that there must be a data flow integrity validation procedure at agent *accountabilityServer* that sanitizes the data flows sent by *localUsers*. The input data flows sent by the *localUsers* are expected to satisfy integrity level 3 and the integrity validation process must check these data flows to guarantee they conform to constraints of integrity level 5. Interpretation of such integrity validation requirement, i.e. what means validation to conform integrity level 5, can be carried out on the basis of dedicated documents such as the specification for data assurance levels by EUROCONTROL [13]. Suitably, this IVF requirement might be implemented by a security mechanism such as a web application firewall that checks for SQL injection/viruses/etc. Thus, as mentioned in Figure 5, at the end of Step1, the security architect obtains the list of security zones, the integrity values of agents and zones and integrity validation filters requirements.

5.2 STEP 2 – Eliciting access control and data flow protection filters requirements

5.2.1 Step2 input

Before running step2, the security architect must complete the output of step1 by providing additional information about the media of communication. The input information concerning media of communication includes the integrity values of the media, the domain in which the media belong to, and finally, the zones connected to them (the white clouds and the black lines in Figure 9).

In our example scenario, we assumed three media of communication: Private access, Public Access and Internet Access. The integrity value attached to a medium of communication represents the level of trust one can have about the packets transmitted by the medium. As consequence, this integrity value depends on the assurance level of the users that can connect to it. The integrity value will be calculated as the minimum between the integrity value of the domain and the connected zones. We also allow the security architect to specify an initial integrity value that represents the risk that unexpected users or attackers have access to the medium. For instance, it is easier to physically access a wireless network than a wired network. Also, it is easier for unwanted users to connect a device to an Ethernet socket in the reception lounge of a building (since many unknown people can enter in this place) than in a restricted area of this building. In this way, our methodology can integrate a second risk analysis phase dedicated to assess the difficulty to physically connect to networks. Figure 10 shows an example of the integrity levels of the media. They are calculated based on the restricted levels of the medium considered in our example scenario. As mentioned in Figure 5, our tool takes two sets of input information at step2: the result of step1, and the media of communication with the associated integrity values when explicitly specified.

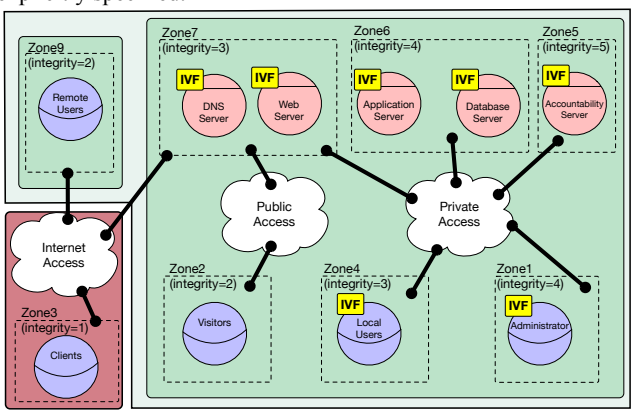

Figure 9. Step 2 - input

Medium	level of Control + openness to access		Restricted level	Integrity
Private Access	Controlled	Restricted to only employees and all the e-commerce server agents	Restricted	3
Public Access	Controlled	Restricted to visitors and WEB/DNS Servers	Partially restricted	2
Internet Access	Uncontrolled	Not applicable	Cannot be restricted	1

Figure 10. Example of medium integrity levels

5.2.2 Step2 – final output.

For the execution of step2, we developed a separate python code that parses the output of step1 and calculates all the paths of every data flows in regards with the consolidated input information at step2. The output of the python code is then re-injected in the ASP solver to compute the final output of step2, i.e., access control filter (ACF) and data flow integrity protection requirements. Figure 11 depicts the graphical representation of the final network security requirements after step2.

The access control filter requirements (ACF) are defined at the interfaces of each zone in order to control all the inter-zone communications. They are depicted by blue squares in Figure 11. These requirements describe the list of permitted flows that are given as input at step1. Data flows not explicitly stated in an ACF requirement are denied. Table 3 shows a sample of the generated ACF requirements. For instance, the requirement statement *ACF(connectedTO(publicAccess,2), flow(dnsServer, visitors))* indicates that there must be an ACF at the interconnection point between medium of communication *publicAccess* and *security zone2* that permits data flows from the *dnsServer* to *visitors*. Depending on the security design choices, these ACFs may be implemented by one or more access control mechanisms such as firewalls, application gateways, etc.

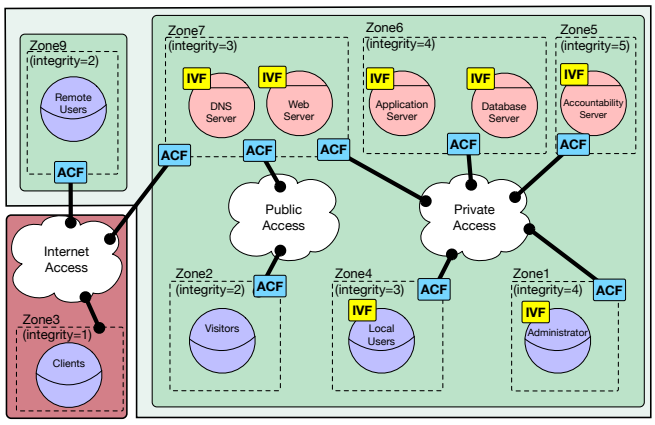

Figure 11. Step 2 output – final network security requirements

Finally, the tool produces data flow integrity requirements representing security protection needs attached to data flows while transiting over a medium or a zone (see Table 4). For instance, requirement *dataFlowIntegrityRequirement(flow(databaseServer, adminUser), privateAccess,4)* states that the data flows from *databaseServer* to *adminUser* must be protected over medium of communication *privateAccess* to ensure the data flows maintain integrity level 4. Different integrity mechanisms such as digital signatures can implement these requirements.

Table 3. Sample of Access Control Filter requirements

```
ACF(connectedTO(publicAccess,2),
    flow(dnsServer,visitors))
ACF(connectedTO(publicAccess,2),
    flow(webServer,visitors))
ACF(connectedTO(publicAccess,2),
    flow(visitors,dnsServer))
ACF(connectedTO(publicAccess,2),
    flow(visitors,webServer))
ACF(connectedTO(publicAccess,7),
    flow(dnsServer,visitors))
ACF(connectedTO(publicAccess,7),
    flow(webServer,visitors))
ACF(connectedTO(publicAccess,7),
    flow(visitors,dnsServer))
ACF(connectedTO(publicAccess,7),
    flow(visitors,webServer))
```

Table 4. Sample of data flow integrity requirements

```
dataFlowIntegrityRequirement(
    flow(databaseServer,adminUser),
    privateAccess,4)
dataFlowIntegrityRequirement(
    flow(appServer,adminUser),
    privateAccess,4)
dataFlowIntegrityRequirement(
    flow(appServer,localUsers),
    privateAccess,3)
```

6 RELATED WORKS

Most of the related works come from industrial/government sectors [10,21,27], which provide best practice guidelines and generic patterns for building secured networks. These guidelines propose some generic zone categories as well as predefined inter and intra zone interactions rules. For instance, the British Columbia model [21] describes seven zones and allows communication inside the zones and only between adjacent zones. Secure Arc [27] defines eight zones. It also introduces a parallel cross-zones segmentation concept, called silos. Communications are allowed only between adjacent zones and within the same silo, or between adjacent silos within the same zone. The aim is to limit the interaction between zones to only dedicated traffic even though they are adjacent to each other. However, these documents are only guidelines and must be manually adapted. As a consequence, they don't support security architects in validating their own network security requirements.

The academic community has published only few works concerning network security zones. Gontarczyk *et al. [16]* proposed a standard blue-print that includes three classes of security zone (no physical measures, limited physical measures, and strong physical measures). It also provides a classifier to guide the deployment of systems/applications. However, this is a high level guideline that must be manually adapted by the security architects. Furthermore, the classifier is ambiguous (e.g. some systems can be placed in any of the zones). They also don't consider other network security requirements. Ramasamy *et al* [25] proposed a bottom-up approach for discovering the security zone classification of devices in an existing enterprise network. Although this work does not deal with eliciting network security requirements, it complements our top-down approach. Several works [17,23] take an existing network security zone model and perform risk analysis using different methods to determine the efficiency of security zones. In the same way, they do not serve the purpose of eliciting high-level network security requirements and cannot be compared to our work. However, they can be used to refine the high-level security requirements calculated by our approach in later stages. Finally, as far as we known, none of the related works consider the notion of cost of network security zones.

7 CONCLUSION AND FUTURE WORK

Network security zone modelling is a well-known approach that contributes to the defense-in-depth strategy from the network security perspective. However, no rigorous approach formally supports this process. In this article, we proposed a zone modelling methodology based on three security principles: complete mediation, least privileges and Clark-Wilson lite formal model. We defined a set of formal rules as well as the list of initial integrity levels values computed based on risk impact, which makes our methodology approach traceable and verifiable. The whole process has been implemented to automate the security zones computation. It produces a set of network security

requirements: security zones, integrity validation filters, access control filters, and data flow integrity requirements. We illustrated the use of this methodology through an e-commerce use case scenario.

Our future works are two folded. First, we plan to formally integrate this work in a global security requirements engineering process to get traceability from business level security objectives to network design level requirements. Secondly, we want to investigate the refinement of the high-level network security requirements produced by our current work. As an example, IVF attached to agents may require to be refined due to design constraints. For instance, it might be impossible to enforce the IVF on the *accountabilityServer* for technical constraints. In this case, the initial security requirements needs to be refined by introducing new security agents (e.g., network security proxies) to achieve the IVF, similar to the final zone model in the case study [2]. In parallel, we would like to extend our security zone modelling approach to consider the confidentiality and availability requirements. Access control filters, defined by our methodology, partially address confidentiality requirements only. We intend to explicitly integrate formal confidentiality models.

ACKNOWLEDGMENTS

This work was supported by the IREHDO2 project funded by DGA and managed by DGAC. We would like to thank our partners from Airbus for their cooperation and their feedback.

REFERENCES

[1] Ross J. Anderson. 2010. *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons.

[2] ANSSI. 2017. Sensibilisation et initiation à la cybersécurité. In *Module 4 : La gestion de la cybersécurité au sein d'une organisation*. Retrieved from https://www.ssi.gouv.fr/uploads/2016/05/cyberedu_module_4_cybersecurite_organisation_02_2017.pdf

[3] Kenneth J. Biba. 1977. *Integrity considerations for secure computer systems*. DTIC

[4] Pierre Bieber, Rémi Delmas, and Christel Seguin. 2011. DALculus–theory and tool for development assurance level allocation. In *International Conference on Computer Safety, Reliability, and Security*, 43–56.

[5] Sravani Teja Bulusu, Romain Laborde, Ahmad Samer Wazan, François Barrère, and Abdelmalek Benzekri. 2016. Towards the weaving of the characteristics of good security requirements. In *International Conference on Risks and Security of Internet and Systems*, 60–74.

[6] Sravani Teja Bulusu, Romain Laborde, Ahmad Samer Wazan, Francois Barrère, and Abdelmalek Benzekri. 2017. Which security requirements engineering methodology should i choose?: Towards a requirements engineering-based evaluation approach. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*, 29.

[7] Sravani Teja Bulusu, Romain Laborde, Ahmad Samer Wazan, Francois Barrère, and Abdelmalek Benzekri. 2018. A Requirements Engineering-Based Approach for Evaluating Security Requirements Engineering Methodologies. In *Information Technology - New Generations* (Advances in Intelligent Systems and Computing), 517–525.

[8] Sravani Teja Bulusu, Romain Laborde, Ahmad Samer Wazan, Francois Barrere, and Abdelmalek Benzekri. 2018. Applying a Requirement Engineering Based Approach to Evaluate the Security Requirements Engineering Methodologies. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing* (SAC '18), 1316–1318. DOI:https://doi.org/10.1145/3167132.3167417

[9] David D. Clark and David R. Wilson. 1987. A comparison of commercial and military computer security policies. In *Security and Privacy, 1987 IEEE Symposium on*, 184–184.

[10] CSEC. 2007. *Baseline Security Requirements for Network Security Zones in the Government of Canada*.

[11] Fabiano Dalpiaz, Elda Paja, and Paolo Giorgini. 2016. *Security requirements engineering: designing secure socio-technical systems*. MIT Press.

[12] Hicham El Khoury, Romain Laborde, François Barrère, Maroun Chamoun, and Abdelmalek Benzekri. 2012. A Formal Data Flow-Oriented Model For Distributed Network Security Conflicts Detection. In *ICNS 2012, The Eighth International Conference on Networking and Services*, 20–27.

[13] EUROCONTROL. 2018. *Specification for Data Assurance Levels*. EUROPEAN ORGANISATION FOR THE SAFETY OF AIR NAVIGATION. Retrieved from https://www.eurocontrol.int/sites/default/files/content/documents/single-sky/specifications/EUROCONTROL%20DAL%20Specification%20Ed%201.1_Released%20Issue.pdf

[14] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2012. Answer set solving in practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 6, 3 (2012), 1–238.

[15] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2014. Clingo= ASP+ control: Preliminary report. *arXiv preprint arXiv:1405.3694* (2014).

[16] Andrew Gontarczyk, Phil McMillan, and Chris Pavlovski. 2015. Blueprint for Cyber Security Zone Modeling. *INFORMATION TECHNOLOGY IN INDUSTRY* 3, 2 (2015), 38–45.

[17] Hannes Holm, Khurram Shahzad, Markus Buschle, and Mathias Ekstedt. 2015. P2CySeMoL: Predictive, Probabilistic Cyber Security Modeling Language. *IEEE Transactions on Dependable and Secure Computing* 12, 6 (2015), 626–639.

[18] ITU-T. 2012. *User Requirements Notation (URN) – Language definition*.

[19] Romain Laborde, François Barrère, and Abdelmalek Benzekri. 2013. Toward authorization as a service: a study of the XACML standard. In *Proceedings of the 16th Communications & Networking Symposium*, 9.

[20] Romain Laborde, Michel Kamel, François Barrère, and Abdelmalek Benzekri. 2007. Implementation of a Formal Security Policy Refinement Process in WBEM Architecture. *Journal of Network and Systems Management* 15, 2 (June 2007), 241–266.

[21] C. Lyons. 2012. *Enterprise IT security architecture security zones: Network security zone standards*. Retrieved from https://www2.gov.bc.ca/assets/gov/government/services-for-government-and-broader-public-sector/information-technology-services/standards-files/network_security_zone_standards.pdf

[22] Microsoft. What is the Windows Integrity Mechanism? Retrieved September 23, 2018 from https://msdn.microsoft.com/fr-FR/library/bb625957.aspx

[23] Robert Mitchell and Elizabeth Walkup. 2017. Further refinements to the foundations of cyber zone defense. In *Military Communications Conference (MILCOM), MILCOM 2017-2017 IEEE*, 823–828.

[24] PRICE. 2015. *True Planning guidance for estimating the cost impacts of ARP-4754, DO-254 and DO-178b/c certification*. Retrieved from https://www.pricesystems.com/Portals/1/Blog/02-05-15-A/DO-178bc%20and%20DO-254%20TP%20Modeling%20Guidance%20-%20DRAFT.pdf

[25] HariGovind V. Ramasamy, Cheng-Lin Tsao, Birgit Pfitzmann, Nikolai Joukov, and James W. Murray. 2011. Towards Automated Identification of Security Zone Classification in Enterprise Networks. In *Hot-ICE*.

[26] J.H. Saltzer and M.D. Schroeder. 1975. The protection of information in computer systems. *Proceedings of the IEEE* 63, 9 (September 1975), 1278–1308. DOI:https://doi.org/10.1109/PROC.1975.9939

[27] SecurArc. Logical Security Zone Pattern. Retrieved from http://www.securearc.com/wiki/index.php/Logical_Security_Zone_Pattern

[28] Adam Sedgewick. 2014. *Framework for improving critical infrastructure cybersecurity, version 1.0*.

[29] Umesh Shankar, Trent Jaeger, and Reiner Sailer. 2006. Toward Automated Information-Flow Integrity Verification for Security-Critical Applications. In *NDSS*.