Integrated management of network and security devices in IT infrastructures.

Bart Vanbrabant, Wouter Joosen
{bart.vanbrabant, wouter.joosen}@cs.kuleuven.be
DistriNet, Dept. of Computer Science,
K.U.Leuven, Belgium

***Abstract*—IT infrastructures just needs to work and at the same time adapt to changing requirements. A significant amount of their downtime is caused by configuration errors and because all other subsystems depend on the network, network errors there have a big impact. Configuration errors are often caused by parameters that are inconsistent because changing one parameter often requires updating multiple other parameters. The configuration of a network is mostly determined by the physical connections and by the subsystems that use the network. Therefore a lot of configuration parameters are derived from other parameters, which increases the risk of inconsistencies. In this paper we present ISIM, a configuration tool for managing a network and its security devices. A first contribution of ISIM is its model with relations between configuration parameters over abstraction and subsystems. This ensures that each parameter only has to be provided once in a configuration and it reduces the work needed to manage a network. The second contribution is a domain model for the network and implementations for this domain model. Other implementations can be added without changing existing configuration code. This enables configuration code reuse. The third contribution is validating the configuration input of ISIM using the types and constraints in the domain model. It reduces the risk of misconfigurations and inconsistencies in the configuration input.**

I. Introduction

Configuring and managing IT infrastructures is a complex problem. The scale of infrastructures keeps increasing and more heterogeneous devices and platforms are added to infrastructures every day. System administrators are faced with an increasing challenge to ensure that all services work as expected. This fact is supported by research that indicates that a significant amount of downtime is attributed to errors caused by human mistakes [1], [2]. Therefore system administrators start to use system configuration tools that automate configuration and management tasks.

It is important for IT infrastructure to always function as intended because most organisations rely heavily on their IT infrastructure. There is no room left for error because downtime is expensive [3], yet IT infrastructures need to evolve constant. Errors are often caused by inconsistencies in the configuration. All instances of a configuration parameter need to be updated and relations between services need to be kept up to date whenever one changes. System configuration tools keep parameters and relations consistent by using parameterised templates [4]. Templates work fine for relations between configuration parameters that are at the same abstraction level. But in computer science abstractions are use to hide complexity and this is no different in IT infrastructures. Relations between configuration parameters span over abstraction levers and parameters at lower abstraction levels are derived from relations at higher abstraction levels [5], [6]. Explicitly modelling all relations removes duplication of configuration parameters, ensuring that a configuration is consistent even when it is frequently updated. It also reduces the amount of work required to manage an IT infrastructure [7]. Unfortunately relations between services spanning multiple abstraction levels cannot be modelled in existing management tools [8].

The networking subsystem of an infrastructure is required by most other services. As a consequence the configuration parameters of the network are determined by the subsystems that use the network and by the physical connections in the network. Only a few parameters can be determined freely by the network administrator. Due to these dependencies an infrastructure can only function correct if the network is secure and functional. Therefore the network needs to be consistent with the services that run on top of it. Network management tools have to be integrated with the tools to manage servers and desktops, so all relations between the network configuration and the configuration of other subsystems can be modelled.

We developed a generic framework for system configuration tools that manage an entire infrastructure. ISIM is a first tool we developed with this framework. It focuses on configuring and managing the network of an infrastructure. In particular, the network configuration of servers and desktops, switches, routers and firewalls. A first contribution of ISIM is configuring the network subsystem using configuration information of other subsystems. It uses the modelling language of our framework to create a configuration model that contains abstractions and relations between these abstractions. This model and the relations in it reduces the number of network parameters that need to be explicitly configured and ensures that the network configuration is consistent with other configuration parameters in the infrastructure. The second contribution is facilitating the reuse of partial configuration models. A configuration model can be factored into modules that separate a configuration model in reusable configuration modules. The third contribution of ISIM is a network domain model that allows verifying the correctness of a configuration model. The domain model contains constraints and types that are used to check if the input model is correct and consistent.

The remainder of this paper is structured as follows: section II provides background on our framework. Section III explains the ISIM tool. ISIM is evaluated in section IV in a realistic network environment. After this section related and future work is discussed in section V and section VI. Section VII concludes this paper.

II. Configuration and management automation framework

Our framework supports configuration tools that manage an entire IT infrastructure. It provides a powerful modelling language with mechanisms to create reusable configuration abstractions that hide the complexity and heterogeneity of the infrastructure. Each abstraction defines an API that is enforced by the framework. The modelling language can be extended with function plug-ins and the language can use templates to generate configuration files. Export plug-ins are available to interface with existing deployment agents or management interfaces. These *artifacts* can be grouped in reusable modules. The modelling language defines a domain model and instantiates a configuration model in terms of the domain model in the same model. Both are integrated to improve the understanding of the entire infrastructure by the system administrator [9]. Templates and function plug-ins transform the input model to a model that can be deployed. Export plug-ins are used to deploy the final configuration model. A conceptual diagram of these components is shown in figure 1.

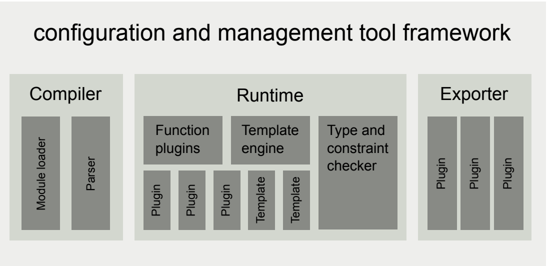

Fig. 1. The components of the framework for developing system configuration tools.

The modelling language is declarative and uses concepts from object orientation [10]. Object orientation is chosen because of its emphasis on modelling the real world and because it is a programming paradigm most IT professionals are familiar with. An example of a model is shown in figure 2. The language contains constructions to model entities using interfaces, relations between entities and instances of these entities. Heterogeneity and complexity are encapsulated inside a construction that is called a class. When an interface is instantiated an implementing class is chosen at runtime.

The modelling language can constrain and validate the input model. There are several mechanisms available to do this in the input model: all data is typed, relations are also typed and have a multiplicity, unique indexes on attributes and relations in an interface and assertions check conditions using Boolean expressions. The most powerful option are plug-ins that register with the runtime to run during the validation phase.

The configuration of most services is done using text files that contain configuration statements. These files exist in all flavours: from simple key-value pairs to xml files or even files that contain commands that are executed. These configuration files are generated using templates that can query the configuration model. Whenever the template engine is not powerful enough to generate complex configuration files from the configuration model the runtime can be extended using plug-ins. These plug-ins can be used to transform the configuration model or to interface with existing management interfaces to enforce the configuration in the infrastructure. Partial configuration models are distributed in modules together with the templates and plug-ins it uses. Modules can be shared or distributed to stimulate *configuration code* reuse. The module mechanism is also useful to split a configuration model up in modules that only contain a domain model and modules with the implementations for this model.

```
typedef mode_t as number matching self >= 0 and self <= 0777

# generic file
native interface File:
    string path
    string content
    string owner
    string group
    mode_t mode
end

interface Host:
    string hostname
    string type
end

Host host [1] -- [0:] File files

index File(host, path)

typedef ConfigFile as File(owner = "root", group = "root", mode = 0644)

class PosixHost implements Host
select: type = "posix"
is:
    # this is a posix host
end

test_host = Host(name = "test.example.com", type = "posix")
motd_file = ConfigFile(path = "/etc/motd",
                content = template("motd.tmpl"), host = test_host)
```

Fig. 2. An example listing of a configuration model in the modelling language

III. Network management using ISIM

ISIM is a system configuration tool to configure and manage a network and its security devices. In real networks only a small number of configuration parameters of the network can be freely determined by the network administrator. Most parameters are already determined by the physical network connections, the properties of the equipment and the requirements of other subsystems in the infrastructure that use the network. ISIM derives the network related configuration from the configuration model of the entire infrastructure. ISIM consists of modules that contain:

- a domain model and the plug-ins that generate configuration parameters from this domain model
- implementation specific modules for specific operating systems or devices

Both types of modules are developed using our framework. Therefore each module consist of input models, templates and plug-ins written in Python.

A. Domain model

Most network configuration parameters are determined by the physical connections and the services that use the network.

The domain model that ISIM provides contains the entities required to model the physical connections, the entities that can be configured by the system administrator and the necessary entities for other tools or modules to model how they use the network.

The types in the domain model related to the physical network provide the interfaces and relations to model the devices (routers, switches and servers), network interfaces and the cables between each interface. The physical links in the model are used by the switch configuration to automatically configure virtual lans. This submodel models the extra parameters that are required to generate a complete network configuration. Most parameters are related to configuring the IP layer of the network. ISIM provides several interfaces to let other modules model how they use the network. For example, interfaces to model network servers and clients. Client and servers connect to each other using a relation. Higher level services use these interfaces to model how they use the network. The domain model also support entities called virtual servers and clients that are only identified with an IP address, IP subnet or IP range. These interfaces are used to complete the configuration model without having to model the entire internet.

Network security policies are modeled in this domain model using a role based model. The policy in ISIM needs to explicitly list all network traffic that is allowed; all other network traffic is denied. Each rule consists of source and destination roles, a service type (protocol and ports) and the direction of the traffic. The input model defines roles that can be linked with interfaces, hosts, IP subnets and the quantifiers used to determine virtual clients and servers. The domain model of this security policy is quite compact and shown in figure 3.

```
typedef direction as string matching self == "one" or self == "both"

interface Role:
    string name
end

Role roles [0:] -- [0:] ip::Network networks
Role roles [0:] -- [0:] net::DefaultVlanInterface interfaces
Role roles [0:] -- [0:] std::Host hosts
Role roles [0:] -- [0:] ip::Ip ips
Role roles [0:] -- [0:] ip::services::VirtualSide virtual

interface Policy:
    string      name
    direction   direction
end

typedef DPolicy as Policy(direction = "one")

Policy policies                 [0:] -- [1:] ip::Service services
Policy source_policies          [0:] -- [1:] Role source
Policy destination_policies     [0:] -- [1:] Role destination
```

Fig. 3. Domain model for security policies. An excerpt from the *fw* module.

B. Implementation

ISIM does not only provide a domain model but also implementations for this domain model. In this paper the implementations are limited to Cisco routers and switches and servers running CentOS 5. New implementations can be added by adding extra modules to the search path of ISIM, without modifying existing modules.

The firewall module verifies if the configuration model complies with the security policy. It also contains the necessary functions to generate rules that can be used by other modules to generate firewall rules for specific firewall implementations. ISIM includes implementations for iptables on CentOS servers and routers and for extended access lists on Cisco routers.

Each service that uses the network has to include the servers, the clients and the connections between them in their model. The firewall module verifies if each connection is allowed according to the firewall policy. If a connection is not allowed; a compilation error is issued. The model verification does not check if a route exists between the server and the client. This is an additional verification step that we would like to include in the future versions of ISIM. ISIM contains the templates and helper functions to generate firewall rules for Linux iptables and Cisco IOS extended access lists. If an instance of *Firewall* is instantiated on a device an implementation for that platform is selected and firewall rules are generated. ISIM generates firewall rules for incoming, outgoing and forwarded connections. Devices that are not configured as router will reject or drop forwarded connections.

ISIM supports Cisco switches that run IOS. The configuration for each switch is generated by checking the connections from each port on the switch. Depending on the configuration on the other end, the port is configured as a trunk port or access port depending on the configuration of the connected device. As a result the configuration of ports on a switch is entirely derived from the configuration of the devices connected to that port. Only management information such as an IP address or time-servers has to be configured explicitly.

C. Higher level services

ISIM also includes modules to manage higher level network services such as DNS. These services are configured transparently based on the network configuration model. The DNS module generates forward and reverse DNS zones for the IP addresses configured in the network configuration model. Each DNS zone is assigned to one DNS server that acts as master and multiple DNS servers that function as slaves for this zone. ISIM generates the master/slave configuration transparently.

IV. Evaluation

In this section we evaluate ISIM by managing a network setup with it. The setup is representative for a network used for hosting a large website. The network contains 15 servers, 2 firewalls, 2 routers and 3 switches. We compare its performance against manually managing the infrastructure by writing the configuration files from scratch and making manual changes. However the network is designed that multiple firewalls are located between different types of services. This is a practice will only be used in networks that have high security demands because of its high maintenance cost. To evaluate the improvements of ISIM over manual configuration we defined metrics based on the lines of code. We compare the lines of code required to manage that infrastructure described using ISIM and the number of lines needed to manually configure

Scenario	initial	adding 4 webservers
Written by user		
Site specific ISIM model	546	558 (+12)
Manual configuration		
Unix config files	5291	5801 (+510)
Cisco config files	894	894
total lines	6185	6695 (+510)

TABLE I
LINES OF CODE IN EACH SCENARIO

the infrastructure. The results are provided in table I. The first column contains the numbers for the initial configuration. The second column are the numbers for a change where four machines that already had a basic configuration are configured as webserver and added to the cluster.

ISIM requires significant less "configuration code" compared to the case where everything is managed manually. Another advantage of using ISIM is that the full configuration is available centrally, instead of having to make changes to the configuration on each machine. Adding four webservers to the cluster makes the advantage ISIM even more apparent. This reduces the effort to make changes and makes it easier to maintain an overview of the infrastructure [9].

V. Related work

The related work for ISIM is both firewall management tools, and network or generic configuration management tools. In "A survey of system configuration tools" [8] we evaluated a set of commercial and often cited generic and network management tools. We identified that the current state-of-the-art provides very little abstraction mechanism and do not allow to model all relations in an infrastructure.

PRESTO [11] is a system configuration tool that focuses on greenfield configuration of very large scale ISP infrastructures. They never model an entire infrastructure but manage each customer separately. Relations over abstractions and over the entire ISP infrastructure cannot be modeled thus creating the risk of inconsistent parameters.

Firmato [12] models the firewall policy of a network and generates firewall configurations for routers in a network. It also provides a tool to visualize firewall rules. FIREMAN [13] also models a security policy and verifies single or distributed firewalls against it. This approach detects anomalies and misconfigurations in the rules of each individual firewall. They both are limited to firewalls and do not integrate with the configuration of the network or the entire infrastructure.

Network configuration management via model finding [14] models the network in a formal model. ISIM uses an object oriented configuration language for this. The main difference is that their approach fills in missing parameters to generate a valid configuration. ISIM will only signal the user that the input model is not complete. ISIM will compile most cases in seconds or in extreme cases minutes. The small example from the paper already [14] requires several hours to compile.

VI. Future work

ISIM is a first step to integrated management of an entire infrastructure. ISIM only manages network and firewall configuration but is built on a framework that can support an integrated management approach. A next step is extending ISIM to manage an entire infrastructure or integrate it in a more generic tool and see how well it scales.

On the network management side ISIM should support routing. Firewall rules are generated without checking if routing is possible. Additionally only static routing is supported and no validation of these routing rules is included. A related improvement is optimising firewall rules and deploying them in an intelligent fashion. Each firewall rule is included in all firewalls, however an internal router possibly cannot support all rules. Instead of deploying no firewall at all, maybe only a limited set of rules could be deployed without creating backdoors.

The metrics used in this limited evaluation have to be extended and measured in multiple change scenarios. An extensive evaluation should also include metrics about the time required to deploy changes, the number of changed lines in an update. Those metrics should also be used to compare ISIM with existing system configuration tools.

VII. Conclusion

ISIM is a tool for managing the network in an infrastructure. Additionally it provides a domain model for other management tools to model how other services use the network. The framework on which ISIM is developed is more generic. Because ISIM consists of only a set of modules for this framework it can be easily integrated in a tool that tries to manage an entire IT infrastructure. Therefore ISIM is a first step to integrate network and firewall management into a generic system configuration tool. This integration ensures that firewalls and network equipment functions exactly how the higher level services expect it to function, resulting in fewer misconfigurations and thus downtime. ISIM reduces inconsistencies by modelling all possible relations between configuration parameters. These relations ensure that if a parameter is updated, all related parameter are also updated. Secondly ISIM separates domain modelling and platform specific implementations. This stimulates reuse of configuration modules and allows for new platforms to be supported without having to make changes to existing configuration libraries. The third contribution of ISIM, is using a typed domain model and constraints to validate the configuration before it is deployed.

VIII. Acknowledgements

This research is partially funded by the Agency for Innovation by Science and Technology in Flanders (IWT), by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy, and by the Research Fund K.U.Leuven.

References

[1] D. Oppenheimer and D. A. Patterson, "Studying and using failure data from large-scale internet services," in *EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop*, ACM. New York, NY, USA: ACM, 2002, p. 255–258.

[2] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do internet services fail, and what can be done about it?" in *USITS'03: Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems*, USENIX Association. Berkeley, CA, USA: USENIX Association, 2003, p. 1–1.

[3] D. A. Patterson, "A simple way to estimate the cost of downtime," in *Proceedings of the 16th USENIX conference on System administration*, USENIX Association. Berkeley, CA, USA: USENIX Association, 11/2002 2002, p. 185–188. [Online]. Available: http://portal.acm.org/citation.cfm?id=1050517.1050538

[4] P. Anderson, *Short Topics in System Administration 14: System Configuration*, Berkeley, CA, 2006.

[5] P. Anderson and E. Smith, "Configuration tools: Working together," in *Proceedings of the 19th Large Installations Systems Administration (LISA) Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 31–38.

[6] T. Delaet and W. Joosen, "High-level system configuration," *The Riseand Riseof the Declarative Datacentre*, pp. 21–22, 2008.

[7] T. Benson, A. Akella, and D. Maltz, "Unraveling the complexity of network management," in *Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, USENIX Association. Berkeley, CA, USA: USENIX Association, 2009, p. 335–348.

[8] T. Delaet, W. Joosen, and B. Vanbrabant, "A survey of system configuration tools," in *Proceedings of the 24th Large Installations Systems Administration (LISA) conference*, Usenix Association. San Jose, CA, USA: Usenix Association, 11/2010 2010.

[9] R. Barrett, P. P. Maglio, E. Kandogan, and J. Bailey, "Usable autonomic computing systems: The system administrators' perspective," *Advanced Engineering Informatics*, vol. 19, no. 3, pp. 213 – 221, 2005, autonomic Computing. [Online]. Available: http://www.sciencedirect.com/science/article/B6X1X-4H758D1-5/2/d6932f83c00847b605187b2694da2b1a

[10] T. Delaet and W. Joosen, "Podim: a language for high-level configuration management," in *LISA'07: Proceedings of the 21st conference on Large Installation System Administration Conference*, USENIX Association. Berkeley, CA, USA: USENIX Association, 2007, p. 1–13.

[11] W. Enck, P. McDaniel, S. Sen, P. Sebos, S. Spoerel, A. Greenberg, S. Rao, and W. Aiello, "Configuration management at massive scale: system design and experience," in *2007 USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, USENIX Association. Berkeley, CA, USA: USENIX Association, 2007, p. 6:1–6:14. [Online]. Available: http://portal.acm.org/citation.cfm?id=1364385.1364391

[12] Y. Bartal, A. Mayer, K. Nissim, and A. Wool, "Firmato: A novel firewall management toolkit," *ACM Trans. Comput. Syst.*, vol. 22, p. 381–420, November 2004. [Online]. Available: http://doi.acm.org/10.1145/1035582.1035583

[13] L. Yuan, J. Mai, Z. Su, H. Chen, C.-N. Chuah, and P. Mohapatra, "Fireman: A toolkit for firewall modeling and analysis," in *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, IEEE Computer Society. Washington, DC, USA: IEEE Computer Society, 2006, p. 199–213.

[14] S. Narain, "Network configuration management via model finding," in *Proceedings of the 19th conference on Large Installation System Administration Conference - Volume 19*, ser. LISA '05, USENIX Association. Berkeley, CA, USA: USENIX Association, 2005, p. 15–15.