COMPUTING SCIENCE

Multilevel Security for Deploying Distributed Applications on Clouds, Devices and Things

Paul Watson and Mark Little

TECHNICAL REPORT SERIES

No. CS-TR-1430 August 2014

TECHNICAL REPORT SERIES

No. CS-TR-1430 August, 2014

Multilevel Security for Deploying Distributed Applications on Clouds, Devices and Things

P. Watson and M. Little

Abstract

The deployment of the components of distributed systems is now often very dynamic - server-side components are virtualised so they can be dynamically deployed on a range of platforms including public and private clouds, while users expect to be able to install clients on devices from phones to tablets. This can introduce security problems that place data at risk. This paper describes a new method for modeling the security of a distributed application and generating the set of possible deployment options that meet the overall security requirements. The model encompasses the entities that influence the security of a distributed system: data, services networks and platforms (e.g. clouds, devices and "things"). The paper describes the method and how it can be used to answer a range of security questions, using a set of case studies including federated clouds, network roaming and "Bring Your Own Devices" (BYOD).

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

Bibliographical details

WATSON, P., LITTLE, M.

Multilevel Security for Deploying Distributed Applications on Clouds, Devices and Things
[By] P. Watson and M. Little
Newcastle upon Tyne: Newcastle University: Computing Science, 2014.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1430)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1430

Abstract

The deployment of the components of distributed systems is now often very dynamic - server-side components are virtualised so they can be dynamically deployed on a range of platforms including public and private clouds, while users expect to be able to install clients on devices from phones to tablets. This can introduce security problems that place data at risk. This paper describes a new method for modeling the security of a distributed application and generating the set of possible deployment options that meet the overall security requirements. The model encompasses the entities that influence the security of a distributed system: data, services networks and platforms (e.g. clouds, devices and "things"). The paper describes the method and how it can be used to answer a range of security questions, using a set of case studies including federated clouds, network roaming and "Bring Your Own Devices" (BYOD).

About the authors

Paul Watson is Professor of Computer Science and Director of the Digital Institute at Newcastle University. He also directs the RCUK Digital Economy Hub on Social Inclusion through the Digital Economy. He graduated in 1983 with a BSc in Computer Engineering from Manchester University, followed by a PhD on parallel graph reduction in 1986. In the 80s, as a Lecturer at Manchester University, he was a designer of the Alvey Flagship and Esprit EDS systems. From 1990-5 he worked for ICL as a system designer of the Goldrush MegaServer parallel database server, which was released as a product in 1994. In August 1995 he moved to Newcastle University. His research interest is in scalable information management with a current focus on Cloud Computing. Professor Watson is a Chartered Engineer, a Fellow of the British Computer Society, and a member of the UK Computing Research Committee.

Dr Mark Little works for Red Hat where he leads JBOSS Technical Direction as well as Research and Development. Following a PhD at Newcastle University, he became Chief Architect and co-founder at Arjuna Technologies. He has been working in the area of reliable distributed systems since the 1980s - his PhD was on fault-tolerant distributed systems, replication and transactions. He holds a position as visiting professor at Newcastle University.

Suggested keywords

CLOUD
SECURITY
DISTRIBUTED SYSTEMS

Multilevel Security for Deploying Distributed Applications on Clouds, Devices and Things

Paul Watson
School of Computing Science
Newcastle University
Newcastle upon Tyne,UK
Email: paul.watson@ncl.ac.uk

Mark Little
Red Hat
INEX Building
King's Road
Newcastle upon Tyne, UK
Email: mlittle@redhat.com

***Abstract*—The deployment of the components of distributed systems is now often very dynamic – server-side components are virtualised so they can be dynamically deployed on a range of platforms including public and private clouds, while users expect to be able to install clients on devices from phones to tablets. This can introduce security problems that place data at risk. This paper describes a new method for modeling the security of a distributed application and generating the set of possible deployment options that meet the overall security requirements. The model encompasses the entities that influence the security of a distributed system: data, services networks and platforms (e.g. clouds, devices and "things"). The paper describes the method and how it can be used to answer a range of security questions, using a set of case studies including federated clouds, network roaming and "Bring Your Own Devices" (BYOD).**

I. Introduction

The components of distributed systems are now far more dynamically deployed than in the past. Server-side components are now often virtualised and dynamically deployed on a range of platforms, including public and private clouds, in order to meet changing performance and cost requirements. Similarly, users want to be able to deploy clients on a diverse set of devices (desktops, laptops, tablets, smartphones). This is increasingly true even for corporate applications.

It has not always been like this. Ten years ago, a typical organisation would deploy its server-side software on fixed machines within its own machine room, while users would be limited to using a very restricted set of clients that were owned and carefully managed by the organisation.

The situation has changed mainly due to user expectations and the need to reduce IT costs. Users now expect access to corporate software applications at anytime, from anywhere and from any device. Recently, users have come to expect that they should be able to access these applications from their own devices: many users now prefer to buy their own devices, such as tablets, and use them for work in preference to a less-attractive standard corporate laptop or desktop. This mobile, "Bring Your Own Device" (BYOD) culture raises a range of challenging security issues, including:

- is it safe to deploy a client on an unknown, user-managed BYOD?
- is it safe for corporate data to be transferred to and from mobile clients over home broadband, coffee-shop Wi-Fi and phone networks?

In parallel with this, there is a move to the more dynamic deployment of virtualised, server-side software components on a range of platforms – including clouds – in order to meet changing demand and reduce costs. This also raises security issues, including:

- which components can be safely deployed on a public cloud?
- which data items can be safely transferred over the Internet to and from a public cloud?

More recently, the introduction of the "Internet of Things" into the distributed systems ecosystem has added another layer of security concerns, with software storing and transmitting sensor data that may be sensitive. For example, data from sensors in the home being transmitted to an application running in the cloud might reveal when the occupants are away.

The typical current approach to addressing this problem is manual and ad-hoc – a human expert will consider a possible deployment plan and decide if it meets the security requirements. This raises numerous concerns:

- the possibility of human error compromising security
- the inability to make very dynamic decisions, for example to understand whether moving a service from a private cloud to a public cloud to handle a performance spike ("cloud-bursting") would break the security requirements
- the lack of an audit trail explaining the decision so that it can be reviewed

This paper addresses these problems by introducing an alternative: a systematic method to model and reason about the deployment of distributed systems in order to meet security requirements. At its heart is a way to formally model the set of entities that determine the security of a distributed system: the services from which it is composed, the data that these services produce and consume, the networks over which the data is transferred, and the platforms on which the services are deployed.

Once the application has been modelled, the paper introduces a method that allows the use of the model for the exploration and validation of the security of an application in terms of these entities. This can be used in two main ways:

- if an administrator specifies the required security levels

for the services and data in an application, the platform on which each service is to be deployed, and the networks it utilises, then the method can determine whether or not the application's security requirements will be met.
- if an administrator specifies the required security levels for services and data, and the range of platforms on which each service could potentially be deployed, the method can generate all deployment options that meet the security requirements (if any exist).

Due to mobility, not all these deployment questions can be answered exclusively at the application's initial deployment time: for example clients may roam across mobile networks. Therefore, there is the need for a new application support framework that enforces the use of the method described in this paper for dynamic decision-making when there are changes in the underlying system, such as network roaming.

The new methods presented in this paper build on our previous work [1] but extend it in significant ways:

- the ability to model networks has been added
- the previous method was restricted to workflows; the new method can be applied to any distributed system, including clouds, clients running on devices from which users access cloud-based applications (e.g. mobile phones, laptops and tablets), and "things" such as sensors in the Internet of Things which transmit data to cloud applications for analysis, or are controlled by those applications
- a new way to model and reason about the security of the system: this combines greater simplicity with increased generality

A tool has been built to implement the method so that users can explore security and deployment options. The tool also automatically generates reports (exploiting LaTeX), and so all the equations, security lattices, results and tables in this paper have been generated automatically in this way.

The paper is structured as follows: it firstly introduces the new model used to represent security requirements; next it shows how that model can be used to represent common distributed system structures; it then shows how the model can be used to answer security questions in a set of case studies involving federated clouds and network roaming. All the case studies are drawn from real-world examples of the challenges faced by industry and researchers wanting to exploit the benefits of cloud computing. Finally, it explains how the method can support an application framework that ensures security requirements are enforced, even in the presence of dynamic changes (e.g. network roaming, or moving services from private to public clouds).

II. A Model for Representing Multi-level Security Requirements

The security model takes into account service, data, device and network security. It is based on the multi-level security models that have dominated security modeling for the past decades, in particular Bell-LaPadula [2].

The entities modeled are:

Notation	Meaning
s_i	Service i (each service has a unique identifier i)
p_i	Platform i (each platform has a unique identifier i)
n_{i-j}	The network connecting platform i to platform j
$d_{i.x-j.y}$	The data sent from service i port x to service j port y
$l(z)$	The security location of z
$c(z)$	The clearance of z (the max l at which z may operate)

TABLE I
Lexical Conventions

Platform
: the underlying hardware and software platforms on which the application is deployed. Examples are the Microsoft and Amazon Clouds, an organisation's data centre, an employee's BYOD cellphone, and a corporate tablet.

Network
: the fixed or mobile networks connecting the platforms on which the application's services are deployed.

Service
: a software component within the application. We model applications as a set of communicating services

Data
: the services communicate by passing data between them

We model applications as a directed graph in which the nodes represent the services, while the edges represent the communications between them. The security requirements of an application are then represented as a conjunction of inequalities that are generated by a set of rules. We now define these rules using the lexical conventions in Table 1.

For each service s_i in the application graph, we add the following inequality:

$$l(p_i) \geq l(s_i) \tag{1}$$

(the security level of the platform on which the service is deployed must be greater than or equal to that of the service.)

For each edge (data connection) $d_{i.x-j.y}$ in the application graph we add the following inequalities:

$$l(p_i) \geq l(d_{i.x-j.y}) \tag{2}$$

(the security level of the platform on which the service transmitting the data is deployed must be greater than or equal to that of the data.)

$$l(p_j) \geq l(d_{i.x-j.y}) \tag{3}$$

(the security level of the platform on which the service receiving the data is deployed must be greater than or equal to that of the data.)

$$l(n_{i-j}) \geq l(d_{i.x-j.y}) \tag{4}$$

(the security level of the network across which the data is transmitted must be greater than or equal to that of the data.)

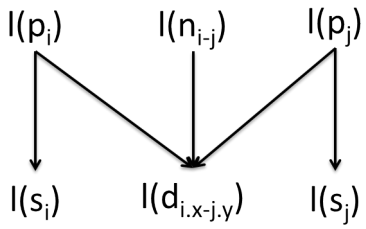

Fig. 1. The Security Level Lattice

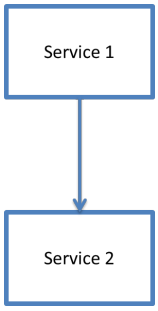

Fig. 2. A producer service sending data to a consumer

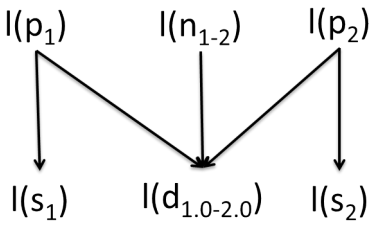

Fig. 3. The Security Level Lattice for a Producer-Consumer application

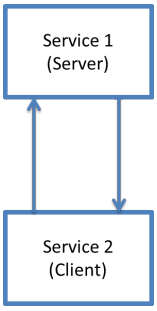

Fig. 4. A Client-Server Architecture

We can also represent these inequalities as shown in the lattice diagram of Figure 1, in which the arrows represent a "$\geq$" relationship.

This method can be used to model any arbitrary application consisting of components that communicate. We now show how these inequalities can be used to model three basic application structures: Producer-Consumer, Client-Server and Pipeline.

1) Application Structure 1: Producer-Consumer: We begin with a simple application involving two services, one of which produces data that is consumed by the other (Figure 2). An example would be that the producer is a sensor, which intermittently sends a reading to a cloud based application – the consumer – that stores and analyses the readings

Applying the above rules to the two services and one data connection allows this application to be modeled as the set of inequalities shown in Equation 5:

$$\begin{aligned} &l(p_1) \geq l(s_1) \wedge \\ &l(p_2) \geq l(s_2) \wedge \\ &l(p_1) \geq l(d_{1.0-2.0}) \wedge \\ &l(p_2) \geq l(d_{1.0-2.0}) \wedge \\ &l(n_{1-2}) \geq l(d_{1.0-2.0}) \end{aligned} \tag{5}$$

We can also represent these inequalities as shown in the lattice diagram of Figure 3.

2) Application Structure 2: Client Server: We now extend application structure 1 to client-server (Figure 4). An example would be where the client is a thermostat that regularly sends temperature readings to a "server" service running in the cloud that stores and analyses these readings in order to decide when to turn the heating on and off. Applying the above rules to the two services and two data connections allows this application to be modeled as the set of inequalities:

$$\begin{aligned} &l(p_1) \geq l(s_1) \wedge \\ &l(p_2) \geq l(s_2) \wedge \\ &l(p_2) \geq l(d_{2.0-1.0}) \wedge \\ &l(p_1) \geq l(d_{2.0-1.0}) \wedge \\ &l(n_{1-2}) \geq l(d_{2.0-1.0}) \wedge \\ &l(p_1) \geq l(d_{1.1-2.1}) \wedge \\ &l(p_2) \geq l(d_{1.1-2.1}) \wedge \\ &l(n_{1-2}) \geq l(d_{1.1-2.1}) \end{aligned} \tag{6}$$

III. Data Security Constraints

So far, we have not considered any constraints on the security relationship between data and services. The method we have presented is extensible and allows for the addition of other security constraints relating to data. This could range from the very basic, in which we impose a simple constraint that the security level of a service must be greater than or equal to that of any data that it reads. Or it could be more sophisticated, for example incorporating the modified Bell-LaPadula method [2] that was introduced for workflows in [1]. We now show how this can be generalised to any distributed system, and represented in the new method presented in this paper.

To model this form of Bell-LaPadula, the following inequalities are added to the security lattice:

For each service s_i in the application graph, we add the

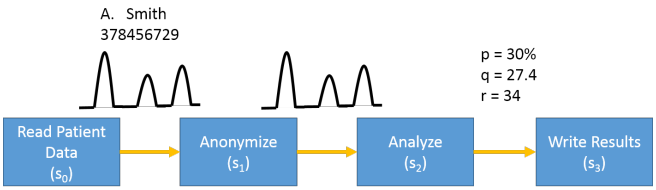

Fig. 5. A Healthcare Workflow for Analysing Sensor Data

following inequality:

$$c(s_i) \geq l(s_i) \tag{7}$$

where $c(s_i)$ represents the *clearance* of the service: its maximum security location

For each edge (data connection) $d_{i.x-j.y}$ in the application graph we add the following inequalities:

$$c(s_j) \geq l(d_{i.x-j.y}) \tag{8}$$

(the Bell-LaPadula "no read up" rule)

$$l(d_{i.x-j.y}) \geq l(s_i) \tag{9}$$

(the Bell-LaPadula "no write down" rule)

IV. Using the Sets of Inequalities to Answer Questions about Deployment

Once the security constraints of the distributed system have been modeled as a set of inequalities, it is possible to answer questions about valid deployment options. We first describe in general terms how this can be achieved; the remainder of the paper then shows how questions from the set of case studies can be answered.

In every case, answering questions takes place in two stages:

1) where there are variables in the inequalities that represent real-world entities whose security levels are known and fixed, bind those variables to those known security levels
2) Simplify the resulting set of inequalities

This can generate one of three results:

- the security constraints can not be met
- the security constraints can be met
- there are specific values (or ranges of values) that the unbound variables can take that would allow the security constraints to be met

Examples of each of these types of results will be given for the case studies presented later. However, we first explain the simplification process.

All the inequalities that are generated are of the form $a \geq b$ where a and b are variables. We represent specific security levels, to which variables can be bound, as integers, with higher values representing higher levels of security. The following rules are then used to simplify the set of inequalities (we use i and j to represent integers, and v to represent variables). Considering first each inequality consisting of a comparison between two integers:

$$i \geq j \tag{10}$$

If this is true then it can be removed from the set of inequalities (as the conjunction of any logical expression E AND $TRUE$ is E); if it is false then the security requirements of the application cannot be met (as the conjunction of any logical expression E AND $FALSE$ is $FALSE$).

Next, consider pairs of inequalities, involving a specific variable v, that are of the form:

$$v \geq i \ \wedge v \geq j \tag{11}$$

These can be replaced by the single inequality

$$v \geq maximum(i, j) \tag{12}$$

Also, inequalities of the form:

$$i \geq v \ \wedge j \geq v \tag{13}$$

can be replaced by the single inequality

$$minimum(i, j) \geq v \tag{14}$$

This process of simplification results in one of: a conjunction of a set of inequalities containing the unbound variables, the discovery that there are no variable values that can be satisfied, or the discovery that the given variable bindings mean that the security requirements have been met. We now show how this process of binding and simplification is used to answer security questions in a set of case studies drawn from examples of attempts to exploit clouds by industry or researchers.

V. Case Studies

This section presents a variety of case studies to illustrate the application and generality of the methodology.

A. Case Study: Federated Clouds

In [1], a method was presented for determining valid deployment options for a healthcare workflow. The aim was to determine the possible ways to partition a workflow across a set of federated clouds so as to meet security constraints. A cost model could then be used to select from the valid options. We will now show that the method of inequalities introduced in this paper can be used to solve this problem more simply than in [1], while also including network security. The workflow is shown in Figure 5 – it is a four service pipeline for analysing accelerometer data. Applying the above rules to the four services and three connections allows this application to be modelled as the set of inequalities. The original paper's model included Bell-LaPadula, but could not encompass network security, which we can now include due to the extended method presented in this paper. The set of inequalities is shown in equation 15, which can also be represented by the security lattice shown in Figure 6.

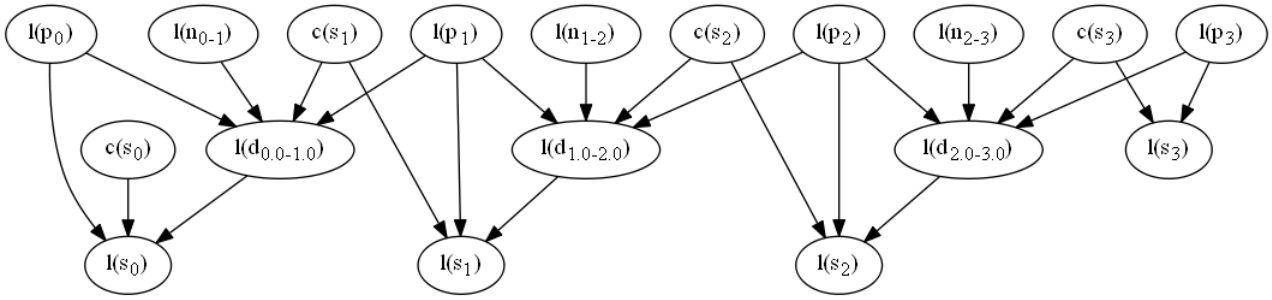

Fig. 6. The security lattice for the Healthcare example, with Bell LaPadula

$$
\begin{aligned}
& l(p_0) \geq l(s_0) \wedge \\
& c(s_0) \geq l(s_0) \wedge \\
& l(p_1) \geq l(s_1) \wedge \\
& c(s_1) \geq l(s_1) \wedge \\
& l(p_2) \geq l(s_2) \wedge \\
& c(s_2) \geq l(s_2) \wedge \\
& l(p_3) \geq l(s_3) \wedge \\
& c(s_3) \geq l(s_3) \wedge \\
& l(p_0) \geq l(d_{0.0-1.0}) \wedge \\
& l(p_1) \geq l(d_{0.0-1.0}) \wedge \\
& l(d_{0.0-1.0}) \geq l(s_0) \wedge \\
& c(s_1) \geq l(d_{0.0-1.0}) \wedge \\
& l(n_{0-1}) \geq l(d_{0.0-1.0}) \wedge \\
& l(p_1) \geq l(d_{1.0-2.0}) \wedge \\
& l(p_2) \geq l(d_{1.0-2.0}) \wedge \\
& l(d_{1.0-2.0}) \geq l(s_1) \wedge \\
& c(s_2) \geq l(d_{1.0-2.0}) \wedge \\
& l(n_{1-2}) \geq l(d_{1.0-2.0}) \wedge \\
& l(p_2) \geq l(d_{2.0-3.0}) \wedge \\
& l(p_3) \geq l(d_{2.0-3.0}) \wedge \\
& l(d_{2.0-3.0}) \geq l(s_2) \wedge \\
& c(s_3) \geq l(d_{2.0-3.0}) \wedge \\
& l(n_{2-3}) \geq l(d_{2.0-3.0})
\end{aligned}
\tag{15}
$$

Next, we bind all the variables to the values from the original paper, leaving only those representing the security locations of the four platforms. These bindings are shown in Table II. The example used only two levels: 0 (representing low security) and 1 (representing high security). The key point about the example is that the workflow first reads a file that contains confidential medical information about a patient, along with the output of a medical sensor. The "Anonymise" service then removes the confidential parts of the file, leaving only the sensor data collected from the patient. This is then analysed to produce a summary of the patient's health. Therefore, while the initial data is confidential (security level 1) and needs to be stored securely, the anonymised data is low security (level 0).

TABLE II
VARIABLE BINDINGS FOR THE HEALTHCARE EXAMPLE

Variable	Location
$l(s_0)$	1
$c(s_0)$	1
$l(s_1)$	0
$c(s_1)$	1
$l(s_2)$	0
$c(s_2)$	0
$l(s_3)$	0
$c(s_3)$	1
$l(d_{0.0-1.0})$	1
$l(d_{1.0-2.0})$	0
$l(d_{2.0-3.0})$	0

TABLE III
VALID OPTIONS FOR RUNNING EXAMPLE

p0	p1	p2	p3
Private	Private	Public	Public
Private	Private	Public	Private
Private	Private	Private	Public
Private	Private	Private	Private

Simplifying the resulting set of inequalities using the rules given above produces the result in equation 16.

$$
\begin{aligned}
& l(p_0) \geq 1 \wedge \\
& l(p_1) \geq 1 \wedge \\
& l(p_2) \geq 0 \wedge \\
& l(p_3) \geq 0 \wedge \\
& l(n_{0-1}) \geq 1 \wedge \\
& l(n_{1-2}) \geq 0 \wedge \\
& l(n_{2-3}) \geq 0
\end{aligned}
\tag{16}
$$

Given the availability of a set of real clouds, with known security levels, equation 16 can be used as the basis for generating all possible valid deployment options as will now be explained.

It is common for organisations to utilise two clouds: one internal "Public" (with security level 0) and "Private" (with level 1). This gives four possible solutions to the inequalities of 16. These are shown in Table III (these solutions were generated by the tool described later in the paper).

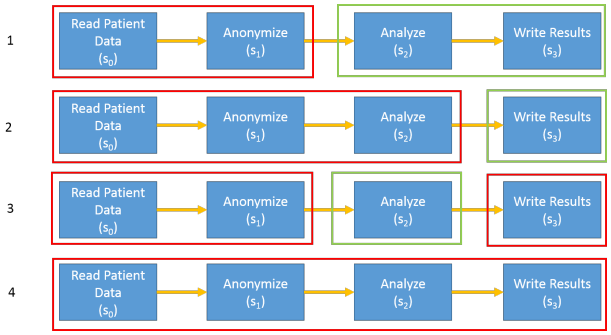

Fig. 7. Options for Partitioning the Healthcare Workflow

Another way to represent this is through the diagram in Figure 7 (the red border surrounds services on the private cloud, the green the public cloud). In order to chose between the options, a cost model such as that in [1] could be used.

Determining all valid options can also be used to assess risks to application availability. Inspection of Table III or Figure 7 shows that the application is entirely dependent on the private cloud: if it fails then there are no options available to run the application sufficiently securely. In response to this knowledge, the organisation may choose to acquire access to a second private cloud to use if the main one fails. If that is the case then the method shows that the options shown in Table IV become available and so the organisation is no longer dependent on a single cloud.

B. Case Study: Can a Bring You Own Device (BYOD) be used as a Client to a Cloud-based Application?

Until recently, organisations concerned with security could exert control over the devices employees used at work by providing managed, controlled and often locked-down hardware and software. This is now changing as employees want to use their personal devices, such as mobile phones and tablets, for work – this is often known as "Bring Your Own Device" (BYOD). Many organisations are concerned by the security implications this creates; in particular, there is the issue of whether trusted data should be accessible from a BYOD. The method described in this paper allows organisations to model and understand such security problems as follows.

The application is modeled as a set of inequalities as in the previous examples, with variables representing the known security levels of the organisation's data, services, platforms and networks bound to the appropriate security levels. The platform variable representing the BYOD would be bound to an appropriately low security value.

The simplification method described above would then be used to determine whether the system, including the BYOD, meets the overall security requirements.

As an example, we take the Client Server application of Section II-2. This was modeled as equation 6. Let us assume that the client in this Client-Server application is running on the BYOD and the server is running in a Cloud. Therefore, p_2 represents the BYOD.

TABLE IV
VALID OPTIONS WITH TWO PRIVATE CLOUDS

p0	p1	p2	p3
Private	Private	Public	Public
Private	Private	Public	Private
Private	Private	Public	Private2
Private	Private	Private	Public
Private	Private	Private	Private
Private	Private	Private	Private2
Private	Private	Private2	Public
Private	Private	Private2	Private
Private	Private	Private2	Private2
Private	Private2	Public	Public
Private	Private2	Public	Private
Private	Private2	Public	Private2
Private	Private2	Private	Public
Private	Private2	Private	Private
Private	Private2	Private	Private2
Private	Private2	Private2	Public
Private	Private2	Private2	Private
Private	Private2	Private2	Private2
Private2	Private	Public	Public
Private2	Private	Public	Private
Private2	Private	Public	Private2
Private2	Private	Private	Public
Private2	Private	Private	Private
Private2	Private	Private	Private2
Private2	Private	Private2	Public
Private2	Private	Private2	Private
Private2	Private	Private2	Private2
Private2	Private2	Public	Public
Private2	Private2	Public	Private
Private2	Private2	Public	Private2
Private2	Private2	Private	Public
Private2	Private2	Private	Private
Private2	Private2	Private	Private2
Private2	Private2	Private2	Public
Private2	Private2	Private2	Private
Private2	Private2	Private2	Private2

TABLE V
EXAMPLE CLIENT SERVER VARIABLE BINDINGS

Variable	Location
$l(p_1)$	1
$l(s_1)$	1
$l(s_2)$	0
$l(d_{2.0-1.0})$	1
$l(d_{1.1-2.1})$	1
$l(n_{1-2})$	1

In this case let us assume that the variables are bound as in Table V. Simplifying equation 6 with these bindings applied produces the inequality:

$$l(p_2) \geq 1 \tag{17}$$

This indicates that the security level of the platform "p_2" must be 1 or greater. Therefore a BYOD assigned a security level of 0 cannot be used as a client in this case.

However, consider another client-server example, identical to V except that the security location of the data flowing in both directions between the BYOD-based client and the cloud-based server is 0. Simplifying the equation representing the system gives:

TABLE VI
BINDINGS FOR THE NETWORK SECURITY EXAMPLE

Variable	Location
$l(p_1)$	1
$l(s_1)$	1
$l(p_2)$	1
$l(s_2)$	1
$l(d_{1.0-2.0})$	1

$$l(p_2) \geq 0 \tag{18}$$

Therefore a BYOD, at security level 0, would be permissible as a client in this case. These examples show that by using the method presented in this paper, an organisation can make considered decisions on whether or not to allow a client to run on a BYOD.

C. Case Study: Network Security

The new method also allows us to explore the security implications of the networks used to transfer data between the services of an application – for example from a service running in a cloud, to a mobile device acting as a client. There are two ways in which this can be done:

1) if the security location of a network is known then the variable representing it can be bound to that value
2) if the security location of a network is unknown then the variable representing it can be left free so that any constraints on its value can be determined

For example, consider the producer consumer system of Figure 2 which was modeled by Equation 5 combined with the variable bindings shown in Table VI. The network security location is not bound, and so simplifying the resulting equation gives the range of possible valid values for the network location. This is shown in equation 19.

$$l(n_{1-2}) \geq 1 \tag{19}$$

Alternatively, if the network location is known then the variable can be bound to this value. For example, if the network location is 0, then simplifying the equation gives the value "False", showing that it is impossible to build a system meeting the security requirements of the data and services using a network with this low level of security.

D. Case Study: Network Roaming

Before the advent of mobile networks, the networks used by an application did not change dynamically and so fixed security location values could be assigned to the variables representing the networks, as in the previous example. With mobile networks comes network roaming, which introduces other issues: the network(s) used by an application might change dynamically while the application is running, for example when a tablet roams away from an organisation's private Wi-Fi to a public cellphone network as the employee moves away from a building.

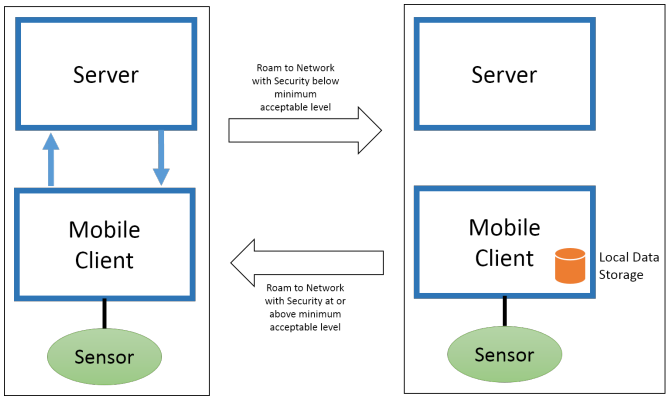

Fig. 8. A Client-Server application that Switches between 2 configurations depending on the Network Secturity Level

We can use the model introduced in this paper to address network roaming. As shown above, it is possible to determine the range of acceptable security location values for the networks used by an application. Then, every time the application wished to roam to use a different network, a check could be made to see if the security location of the new network was known to be in the acceptable range (we assume the existence of a table mapping known networks to security levels: any network not in the table could be assumed to be of the lowest security level). If the network was in the acceptable security range then the application could be allowed to continue to operate. If not then the application could react in one of two ways:

1) the application could pause operation until another network, with a sufficiently high level of security, became available
2) if it was possible, the application could adapt to a (temporary) mode of operation which did not require the use of the insecure network.

An example of the second case is shown in Figure 8. An application contains a mobile client that regularly reads from a sensor and sends it to a server for storage and analysis. This is shown on the left side of the diagram. However, the data is sensitive, and so requires a network connecting the services that operates at a high level of security. If roaming takes place to a network that is not at this level then the application could switch into the configuration shown on the right side of the figure, in which the client reads and caches locally the data from the sensor. When the mobile client roams to a sufficiently secure network then the cached data is sent to the server, and the application switches back to the configuration on the left side of the diagram in which data read from the sensor is sent straight to the server. This is equivalent to many mobile phone apps which are designed to work both on and off line, however the method described in this paper has the advantage that it can inform the designer that the application is sensitive to the network security level, and the tool implementing the method can be used in the code executed during roaming to determine whether a switch to another configuration of the application

is required.

VI. Implications for System Design

Deploying this method for real-world distributed systems requires:

1) a tool to generate the structure of an application, either automatically or with manual assistance
2) a tool to allow the person responsible for the security of an application to specify the security requirements for both data and services
3) a tool to allow a security expert to specify the security levels of platforms and networks
4) a tool to use the method described in this paper to check prior to the deployment of an application whether or not the proposed initial deployment plan (including specific platforms and networks) meets the security criteria
5) prevention of automatic roaming where the security of an application may be compromised. A check should be introduced to ensure that the new network meets the security requirements before roaming is allowed to take place. If it does not, then the application must either be halted (until a sufficiently secure network is available) or adapted so that it can continue without using the low security network (as described above). The application may adapt back once a sufficiently high security network becomes available.

We have written a tool (in Haskell) to support the implementation of items 1-4 above. This includes automatically creating the set of inequalities representing a system, binding variables, simplifying the set of inequalities representing the system, and generating all possible deployments onto real-world entities such as clouds. All the examples from this paper have been implemented in the tool. The tool can also generate its outputs in the form of LaTeX reports, tables and equations for scrutiny by an expert: all the tables, lattice diagrams, equations and results in this paper have been produced in this way. The GraphViz library is used for generating diagrams [3] in the reports.

Fully supporting item 4 requires action at deployment time. When a platform requests the deployment of a service within an application (e.g. to deploy a client on a BYOD or a server-side component on a public cloud), the application deployment system must determine if this is allowed according to the security criteria of the application. This requires a method by which platforms can identify themselves (e.g. through a mac address or a protected identifier). The deployment system would then use a table of known devices to map the device identifier to a platform security level. If the device was unknown then the lowest security level would be assigned. When the platform security level had been discovered, the security model of the application could determine if the service was allowed to be deployed on that device. This is analogous to the way in which some app stores for mobile devices prevent apps from being deployed on "rooted" devices.

To support item 5 above (network roaming) requires that communication between services is mediated by a system that detects when roaming is about to occur, looks up the security level of the proposed new network in a table (defaulting to the lowest security level for unknown networks), and then uses a tool implementing the method introduced in this paper to determine whether the proposed new network has a sufficiently high security level. If so, then roaming is allowed, if not then the application either pauses, or adapts to an off-line mode of working as described above in sub-section V-D.

VII. Related Work

This paper is a major development of the ideas found in an earlier paper [1], which took a similar, but more restricted approach and applied it to cloud security. Unlike the new method, it was unable to handle general distributed systems, nor networks. The method in [1] was further formalised in [4].

Our motivation for working on cloud security is the result of trying to find a systematic method to deal with the security issues we encountered in our experiences of exploiting clouds for industrial and research applications. However, ever since clouds first appeared, security has been a general concern that has prevented many organisations from exploiting the benefits of the cloud for at least some applications [5]. An overview of cloud security concerns and possible solutions is given in [6]. It also describes the use of trusted security entities and a security broker that can match risk assessments to requirements.

The method presented in this paper requires the assignment of security levels to platforms, services, data and networks. Ways of achieving this are discussed in [7] and [8].

Bell-LaPadula was also applied to workflow security in [9]. The underlying formalism used was Petri Nets, unlike the approach taken in this paper. It also had the restriction that it modelled a form of Bell-Lapadula that included *clearance* but not *location*. Further, the paper did not consider the deployment of services across a set of computational resources, as that was not in its focus.

Recently, there have been other papers discussing the allocation of services to cloud resources in order to meet security requirements. [10] describes a method of partitioning workflow over clouds. It incorporates the security constraints of [1], and adds others (e.g. for keeping entities together or apart); it also introduces a method for optimising the placement of partitions on clouds to meet QoS requirements. It limits its focus to workflow (rather than general distributed systems) and does not consider network security or mobility.

VIII. Conclusions

We believe that it is important to have a formalised, auditable method for reasoning about security decisions relating to deploying distributed systems across a range of platforms. This is increasingly important given the wide range of platforms now available, ranging from highly scalable clouds (both public and private), through mobile devices, to the sensors ("things") that are now proliferating. Before we devised the method in this paper, and its more limited predecessor [1] we often found ourselves having to make manual judgements

about how to partition complex, sensitive applications across a set of platforms (often public and private clouds). The consequences of making an error could have been great in terms of regulation and trust. For example, some of our projects involve healthcare applications in which placing patient data at risk could have severe consequences. This could be due to sending sensitive data over insecure networks, storing it on insecure platforms, or allowing untrusted, low-security services to operate on it. We therefore set out to design a method to avoid this.

In this paper we have described this new method for modelling the security requirements of distributed systems that run on multiple platforms, along with a way to determine valid options for deploying the components of the distributed system. We believe the approach to be completely general, and have shown its utility through a set of case studies drawn from real applications in industry and research involving clouds, devices and "things". One major advance over the earlier work in [1] is in the addition of the ability to model networks. The motivation for adding this capability was a use-case from industry involving the security of systems with clients running on mobile devices – focussing solely on the back-end of systems running on clouds was no longer sufficient to ensure system security. Once we had proved the ability to encompass networks, we were then faced with a real-world scenario involving a roaming client.

This led to the idea described in Section VI in which roaming from one network to another could trigger a transformation from one system configuration to another. Achieving this requires the ability to run the model when the network changes to determine if a change in system configuration is needed – this requires the model to be implemented in the form of a tool, which we have built, that can be executed at run-time to check the overall system security levels.

All the equations and tables in this paper, as well as the security lattice of Figure 6 are generated by the tool, which reduces the risk of mistakes being made due to transcription errors. When combined with a cost model (as in [1]) the tool can also be used to automatically generate, deploy and execute the partitions of the distributed applications on a set of platforms. We have implemented and evaluated this for e-Science Central – an open-source cloud workflow platform [11]. Our current research is in integrating this work with exception-handling and fault recovery in order to deal efficiently with cases where clouds fail [12], or become available, during the execution of an application.

ACKNOWLEDGMENT

This work was funded by the Research Councils UK Social Inclusion through the Digital Economy project EP/G066019/1 and by Red Hat. The authors would also like to thank Hugo Hiden, Simon Woodman, Jacek Cala, Zhenyu Wen, Stuart Wheater, Stian Thorgersen, Sabina Fataliyeva, Polina Zablotskaya and Vera Miloslavskaya for their helpful comments and suggestions.

REFERENCES

[1] P. Watson, "A multi-level security model for partitioning workflows over federated clouds," *Journal of Cloud Computing*, vol. 1, no. 1, pp. 1–15, 2012.

[2] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," MITRE Corporation, Tech. Rep., Mar. 1973.

[3] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz open source graph drawing tools," *Graph Drawing*, vol. 2265, pp. 483–484, 2002. [Online]. Available: http://www.springerlink.com/index/bvdkvy7uj1plml8n.pdf

[4] L. Freitas and P. Watson, "Formalizing workflows partitioning over federated clouds: multi-level security and costs," *International Journal of Computer Mathematics*, no. ahead-of-print, pp. 1–26, 2013.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[6] O. Wenge, M. Siebenhaar, U. Lampe, D. Schuller, and R. Steinmetz, "Much ado about security appeal: cloud provider collaborations and their risks," in *Service-Oriented and Cloud Computing*. Springer, 2012, pp. 80–90.

[7] S. Pearson and T. Sander, "A mechanism for policy-driven selection of service providers in SOA and cloud environments," in *New Technologies of Distributed Systems (NOTERE), 2010 10th Annual International Conference on*, 31 2010-june 2 2010, pp. 333 –338.

[8] J. Mace, A. van Moorsel, and P. Watson, "The case for dynamic security solutions in public cloud workflow deployments," in *The First International Workshop on Dependability of Clouds, Data Centers and Virtual Computing Environments (DCDV 2011)*, Hong Kong, China, 2011.

[9] K. Knorr, "Multilevel security and information flow in Petri Net workflows," in *Proceedings of the 11th Conference on Advanced Information Systems Engineering*, 2001.

[10] E. Goettelmann, W. Fdhila, and C. Godart, "Partitioning and cloud deployment of composite web services under security constraints," in *Cloud Engineering (IC2E), 2013 IEEE International Conference on*. IEEE, 2013, pp. 193–200.

[11] P. Watson, H. Hiden, and S. Woodman, "e-science central for CARMEN: science as a service," *Concurr. Comput. : Pract. Exper.*, vol. 22, pp. 2369–2380, December 2010. [Online]. Available: http://dx.doi.org/10.1002/cpe.v22:17

[12] Z. Wen and P. Watson, "Dynamic exception handling for partitioned workflow on federated clouds," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1. IEEE, 2013, pp. 198–205.