Ariadne: Topology Aware Adaptive Security for Cyber-Physical Systems

Christos Tsigkanos[†], Liliana Pasquale[*], Carlo Ghezzi[†], and Bashar Nuseibeh[*‡]
[†] Politecnico di Milano, Milano, Italy
[*] Lero - the Irish Software Research Centre, Limerick, Ireland
[‡] Department of Computing, The Open University, Milton Keynes, UK

***Abstract*—This paper presents Ariadne, a tool for engineering topology aware adaptive security for cyber-physical systems. It allows security software engineers to model security requirements together with the topology of the operational environment. This model is then used at runtime to perform speculative threat analysis to reason about the consequences that topological changes arising from the movement of agents and assets can have on the satisfaction of security requirements. Our tool also identifies an adaptation strategy that applies security controls when necessary to prevent potential security requirements violations.**

I. Introduction

Ubiquitous computing is resulting in a proliferation of cyber-physical systems (CPS) that host or manage valuable physical and digital assets. These assets can be harmed by malicious agents through both cyber-enabled or physically-enabled attacks, particularly ones that exploit the often ignored interplay between the cyber and physical world. For example, cyber-enabled physical attacks can occur when physical assets are cyber-controlled, such as software that controls access to buildings or some of their areas. Similarly, physically-enabled cyber attacks can occur when physical access to assets or locations enables cyber attacks. For example, access of an agent to a physical area from which it is possible to connect to a wifi network, enables the agent to eavesdrop on other agents' information transmitted over the network.

In previous work [1], [2] we advocated that the topology of cyber and physical spaces -their structure in terms of key elements and their relationships- can provide a system with awareness of security relevant contextual characteristics. These include the location of assets being protected and the security controls that should be enacted in their proximity. Moreover, the location of potentially malicious agents can increase the threat of harm to the assets in their vicinity. Discovering threats arising from the interplay of cyber and physical spaces suggests the need for an explicit representation of the topology of such spaces including properties such as structure and connectivity. Changes in the topology triggered by movements of objects or agents in the physical or cyber space can also change the attack surface dynamically, and introduce new threats that are difficult to identify at design time. Existing approaches [3] do not discover and counter security breaches at runtime determined by unexpected topological changes of the cyber-physical space; an adaptive security approach to discover and mitigate possible security threats is needed.

This paper presents Ariadne, a tool for engineering topology aware adaptive security for cyber-physical systems. This prototype tool demonstration embodies our approach in the context of *known unknowns*, where although change primitives are well understood by the security engineer, their effect on security requirements at runtime cannot be predicted. Ariadne allows defining and maintaining a live model of the topology of cyber and physical spaces and performs speculative threat analysis and planning, to respectively discover and counter security threats at runtime. Potential users are security software engineers who have to manually configure security controls in cyber-physical environments, such as smart buildings. Ariadne is demonstrated through two adaptive security scenarios set in a smart building, involving respectively a physical and a cyber threat both enabled by changes in the topology of the physical space. The rest of the paper illustrates the steps necessary to engineer topology aware adaptive security, describes the demonstration scenarios and explains how the functionalities provided by Ariadne can identify and prevent identified security threats.

II. Demonstration Scenarios

As an example consider a floor plan of a modern building, comprising a corridor, from which a server room and a wifi area are accessible. The floor plan is illustrated in Figure 1. Wireless network access is provided for the agents located in the wifi area. The server room contains valuable assets, such as a printer, a server and a cloudlet, provisioning virtual machines upon request to users connected to the wireless network. Communication with virtual machines allocated in the cloudlet can also be encrypted if necessary. Agents can roam into the building, while also carrying their mobile phones; access to areas is regulated by authenticating on room doors.

As security requirements we consider that external visitors should always be accompanied in the server room, in order to preserve the integrity of assets inside, such as the server (SR1). Furthermore, information transmitted by an agent to one of the VMs allocated on the cloudlet should never be received by others (SR2). Enabling predetermined security controls to satisfy the aforementioned security requirements might not be effective as topological changes arising from the movement of agents and assets can still lead to security breaches. For

We acknowledge SFI grants 10/CE/I1855, 13/RC/2094 and ERC Advanced Grants (ASAP) no. 291652 and (SMScom) no. 227977.

example, preventing visitors from accessing the server room might no longer be effective when for example, a visitor requires access to the server room to fix the printer. Similarly, always enabling encryption on mobile devices communicating with the cloudlet is not efficient as they can run out battery.

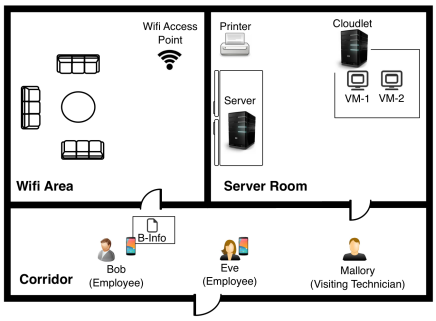

Fig. 1: Floor plan of a building

For the first demo scenario, we consider a visiting technician, Mallory, who requires access to the server room to repair the printer. If she is unaccompanied, requirement SR1 is violated. To prevent this security breach several actions can be performed while Mallory is in the corridor. For example, she can be forbidden from accessing the server room until another employee (e.g., Bob) has accessed the server room. For the second scenario, we assumed an employee (e.g., Bob) is in the wifi area and communicates with one of the VMs created on the local cloudlet. If another employee (e.g., Eve) moves to the wifi area while carrying a mobile device and connects to the wireless network, she can eavesdrop on Bob's information transmitted over the network. To prevent this security breach Eve can be forbidden from accessing the wifi area or from connecting to the network; alternatively encrypted information transmissions can be enabled.

III. Topology Aware Adaptive Security

Figure 2 provides an overview of our topology aware adaptive security approach. Adaptation builds on a live representation of the topology of the cyber and physical space characterising a system operational environment modelled using *Bigraphical Reactive Systems* (BRS) [4]. Adaptive security is achieved by implementing the activities of the MAPE (Monitoring, Analysis, Planning and Execution) loop. Events generated from the cyber and physical space indicate the execution of agents' actions.

During *monitoring*, events indicating the execution of agents' actions are received. During speculative threat *analysis*, future configurations of the topology of the cyber and physical space that can be achieved when agents perform actions are looked ahead up to a certain number of steps. Violations of security requirements that can arise from future topological configurations are subsequently identified through model checking. The analysis requires as input the updated model of the cyber and physical space and the security requirements. After analysis terminates, a set of action execution traces leading to violations is generated.

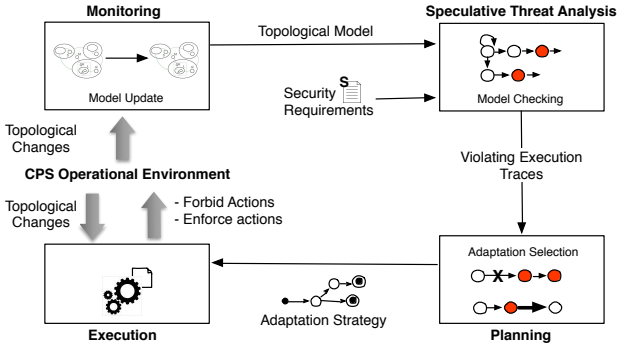

Fig. 2: Topology aware adaptive security

Action execution traces leading to violations are used during *planning* to determine an adaptation strategy aiming to prevent future violations of security requirements. An adaptation strategy can forbid the execution of some actions that lead to a violation or can enforce additional actions to restore a violated security requirement. Selection among more than one actions that can be enforced depends on their cost which is predetermined by the security software engineer. During *execution*, the adaptation strategy is recognised and applied on the real cyber physical system. This activity receives as input the events indicating the execution of agents' actions, identifies when a specific state in the adaptation strategy is reached, and enforces an adaptation action if necessary.

IV. Modelling Cyber and Physical Spaces

Ariadne allows security software engineers to manually define a topological model of the operational environment, a set of reaction rules, and the security requirements. The model of the topology is expressed using bigraphs [4], which is a formalism consisting of two graphs. A *place graph* is a forest, capturing the notion of containment and nesting of physical and digital locations. A *link graph* is a hypergraph composed of the same set of nodes of the place graph and a set of edges each linking any number of nodes; this graph represents generic many-to-many relationships among nodes, such as communication among digital objects and agents.

$P.Q$	*Nesting* (P *contains* Q)	(1a)
$\$i$	*Site numbered* i	(1b)
$K[a,b].P$	*Node associated with control* K *having ports* a *and* b. *The node contains* P.	(1c)
$P \parallel Q$	*Juxtaposition of roots*	(1d)
$P \mid Q$	*Juxtaposition of children*	(1e)

The bigraph representation used for the scope of this tool is found in Formulae 1a-1e, and is used to represent the floor plan of the demonstrating example, as shown in Listing 1. P and Q are bigraphs. The containment relationship is expressed in Formula 1a; for example, $Agents$ eve and bob contain a mobile phone they carry. Additionally, bigraphs can contain sites (Formula 1b) that can be used to denote placeholders. Controls are names that define a node's type; each node control

can be associated with a number of named ports. If a single instance node of that type exists in the bigraph, the control also uniquely identifies that node (e.g., for the building). Otherwise, we use port names as a way to uniquely identify a node (e.g., for rooms, visitors, or other agents). Since the underlying link graph is a hypergraph, multiple different ports may be related to a single name; ports of different nodes occuring in a formula with the same name, are considered connected. For example, $Phone$s can have port $wlink$ indicating they are connected to the wireless network (note the $Wifi$ router having also a port named $wlink$). Bigraphs can be contained in roots that delimit different hierarchical structures. In Formula 1d, P and Q are different roots; they can reside in different parts of the containment hierarchy. Conversely, two bigraphs can be placed at the same hierarchical level , as shown in Formula 1e. For example, nodes representing the printer and the server are co-located in the $Room$ identified by port $serverroom$.

Listing 1: Representation of the example floor plan

```
Building.( Room[corridor].(
  Room[serverroom].( Printer|Server|Cloudlet[I])
  | Visitor[mallory] | Agent[eve].(Phone[-])
  | Agent[bob].(Phone[-].(Info[bob])) |
  Room[wifiarea].(Wifi[wlink,I]) ))
```

BRS extend bigraphs by representing the dynamic behavior of the operational environment as a set of reaction rules. These rules have the general form of $R \rightarrow R'$, where R is a redex and R' is a reactum; both the redex and reactum are bigraphs. In particular, if a part of a bigraph matches a reaction redex, it can be replaced with the reactum, in a fashion similar to graph rewriting.

TABLE I. BRS REACTION RULES

Reaction rule
enter_room
`Agent[x].($0) | Room[r].$1 | $2 ->` `Room[r].($1 | Agent[x].$0) | $2`
connect_wifi
`Wifi[wlink,I].$1|Agent[x].Phone[-].$2|$0 ->` `Wifi[wlink,I].$1|Agent[x].Phone[wlink].$2|$0`
create_vm

`Wifi[wlink,I].$4 || Agent[x].Phone[w].$2 ||`
`Cloudlet[I].$3 ->`
`Agent[x].Phone[wlink].$2 ||`
`Wifi[wlink,I].$4|| Cloudlet[I].(VM[x]|$3)`

info_tx

`Agent[x].Phone[w].Info[x] ||`
`Agent[b].Phone[wlink].$5||Wifi[wlink,I] ||`
`Cloudlet[I].(VM[x] |$3) ->`
`Agent[b].Phone[wlink].(Info[x]|$5)`
`|| Agent[x].Phone[wlink]||Wifi[wlink,I]`
`|| Cloudlet[I].(VM[x].(Info[x])|$3)`

Table I represents some of the reaction rules modeled for our demonstration scenario. Rule $enter\_room$ represents an $Agent$'s ability to enter a room; if an $Agent\ x$ is at the same hierarchical level with $Room\ r$, she can access it, while all the other elements contained either in the $Agent\ x$, around or in the adjacent $Room\ r$ are not modified. In the same fashion, connection to the wifi network is modelled by joining the mobile phone carried by an $Agent$ to port $wlink$ already connected to the $Wifi$ router. Similarly, rule $create\_vm$ expresses the capability of an $Agent$ containing a $Phone$ which is connected to the $Wifi$ (i.e. it has port $wlink$), to allocate a VM on the $Cloudlet$. Information transmission is modelled as the exchange of a token; for unencrypted transmissions reaction rule $info\_tx$ models the fact that this information token may end up also on another $Agent$'s phone connected on the $Wifi$ network if encryption is not enabled; conversely if encryption is used, information is regarded as transmitted securely to the recipient.

Security requirements are represented as parametric matching properties, themselves as bigraphs. To match, two bigraphs have to exhibit the same structure as well as the same link connections. The security requirements of the demo scenario are represented in Table II. For example, the physical integrity (SR1) of assets placed in the server room is violated when a $Visitor$ is located inside without an accompanying $Agent$; a visitor is allowed alone only in the corridor. Moreover, the requirement corresponding to a violation of an $Agent$'s confidentiality (SR2) can be represented as an $Agent$ whose $Phone$ appears to carry an information token associated with a different $Agent$.

TABLE II. SECURITY REQUIREMENTS

Name	Requirement

Integrity	`Room[corridor].(Visitor[x]|$0)|$1 OR`
	`Room[serverroom].(`
	`Visitor[x]|Agent[y]|$2)|$3`

Confidentiality	`Agent[eve].Phone[wlink].(` `Info[bob]|$0)|$1`

V. A WALK THROUGH ARIADNE

Ariadne maintains an updated representation of the topology at runtime. It receives events corresponding to the execution of actions in the environment; each event is associated with a reaction rule in the BRS model. Events indicate the action that occured, its enactor agent(s) and a subject if applicable; e.g. $\langle bob, enter\_room, wifi\_area \rangle$ indicates that Bob has accessed the wifi area. The model of the cyber-physical space maintained at runtime is updated, reflecting the application of reaction rules associated with the monitored events.

The updated BRS model is subsequently used to perform speculative threat analysis. To achieve this aim, a Labelled Transition System (LTS) [5] is generated representing possible future topological configurations of the cyber-physical space. This exhaustive state generation process is bounded by a specific look-ahead depth, indicating the maximum number of subsequent actions that will be considered during model checking; such a look-ahead depth can be configured by the security software engineer depending on runtime performance constraints. Each LTS state represents a different topological configuration of the cyber-physical space, whereas each LTS transition connects two configurations and indicates that the

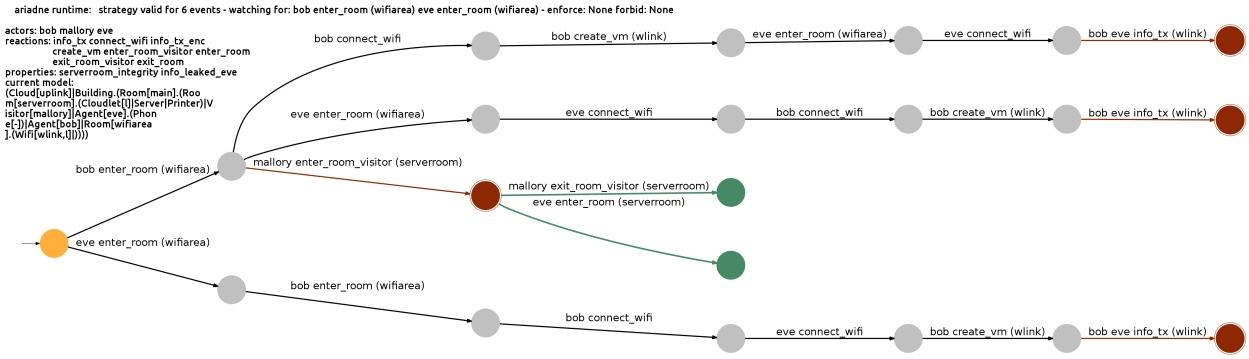

Fig. 3: Screen of Ariadne runtime showing a fragment of an adaptation strategy

target state is reachable from the starting one, when a specific reaction rule is executed. Different sequences of transitions can lead to the same topological configuration.

The tool presented in this paper uses a set-theoretic approach to generate new states in a Bigraphical Reactive System, which then maps to a LTS and proceeds to perform model checking. At each state generated, the security requirements specification is checked. Should a violation occur, the complete counterexample is considered. For example, as shown in Figure 3, if Bob enters the wifi area and Mallory accesses the server room, requirement SR1 is violated (red state) as Mallory is unaccompanied. A set of immediate successor states of the violating one in which the security property is restored are also identified (green states). For example, for the scenario where Mallory enters the server room unaccompanied, if subsequently Eve enters the server room or Mallory is forced to exit, integrity is restored again.

The execution traces generated by model checking are used to compute an adaptation strategy valid for a specific number of subsequent topological configurations of the cyber-physical space. Such a strategy forbids the execution of actions that lead to the violation of a security requirement, or alternatively perhaps enforces specific actions in order to mitigate it. For this purpose a state machine is generated that recognises the transitions of the counterexamples received as input from the analysis, as these are the ones that lead to violations. This state machine activates appropriate adaptation actions if necessary.

A screenshot of the runtime environment showing a limited fraction of the adaptation strategy generated is displayed in Figure 3. In the upper left part, various runtime information of the prototype tool is provided; these include the current bigraphical model, the last event received and the actions taken at the current state (shown in yellow). Additionally, the lookahead depth used for the analysis of the state space and the events the system is waiting for to progress the adaptation strategy. If we consider the state reached when Bob enters the wifi area, connects to the network and creates a VM, if subsequently Eve enters the room, and connects to the network and Bob transmits information to one of his allocated VMs, a state is reached where confidentiality is violated. Indeed, Bob's information is transmitted insecurely and therefore it is considered as also received by Eve. To avoid such a violation unencrypted transmission is forbidden after Eve connects to the wifi network. Decisions on which actions can and are in fact enforced or forbidden are predefined by the security software engineer depending on the application domain. For example, if the system is unable to enforce encrypted connections but could control access to rooms, Eve could be prevented from entering the wifi area instead.

A. *Demonstration Testbed*

The test-bed[1] we developed to reproduce the cyber-physical system of the demonstration scenario includes an OpenStack installation simulating a cloudlet and exposing a service for dynamically creating, deleting and using VMs, a wifi access point (OpenWrt), NFC readers associated with the doors of the wifi area and the server room, and smartphones used to connect to the wireless network. An Android application was also developed to create/delete VMs and issue requests to them from mobile devices through the wifi network. This application also supports conditional encryption of messages exchanged between the mobile phone and the cloudlet. Monitored events transmitted to Ariadne included: access to a room by an agent, connection/disconnection of a mobile device to/from the wireless network, provisioning/deprovisioning VMs on the cloudlet, and message exchanges between a mobile phone and a VM. As adaptation actions we allowed the possibility to forbid an agent from accessing a room or disable unencrypted communications between mobile phones and the cloudlet.

REFERENCES

[1] L. Pasquale, C. Ghezzi, C. Menghi, C. Tsigkanos, and B. Nuseibeh, "Topology Aware Adaptive Security," in *SEAMS*, 2014, pp. 43–48.

[2] C. Tsigkanos, L. Pasquale, C. Menghi, C. Ghezzi, and B. Nuseibeh, "Engineering Topology Aware Adaptive Security: Preventing Requirements Violations at Runtime," in *RE*, 2014.

[3] L. Pasquale and C. Tsigkanos, "A review of formalisms and analysis techniques for cyber-physical systems," Tech. Rep. [Online]. Available: http://home.deib.polimi.it/tsigkanos/techrep/cps.pdf

[4] R. Milner, *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model checking*. MIT press, 1999.

[1] A video demonstration can be found at http://youtu.be/Z8ODO96jGMw