Ensuring Security and Privacy in Social Virtual Reality Learning Environments

Samaikya Valluripally[1], Aniket Gulhane[1], Khaza Anuarul Hoque[1], Reshmi Mitra[2], and Prasad Calyam[1]

[1] University of Missouri-Columbia, Missouri, USA
{svbqb, arggm8,}@mail.missouri.edu, {hoquek, calyamp}@missouri.edu
[2] Webster University, St.Louis, USA
reshmimitra25@webster.edu

Abstract. Social Virtual Reality Learning Environment (VRLE) is a novel edge computing platform for collaboration amongst distributed users. Given that VRLEs are used for critical applications (e.g., special education, public safety training), it is important to ensure security and privacy issues. In this paper, we present a novel attack tree model with formalism and model checking techniques to obtain quantitative assessments of threats and vulnerabilities. Based on the use cases from an actual social VRLE viz., vSocial, we describe security and privacy attack trees that when converted into a stochastic timed automata allows for rigorous statistical model checking. Such an analysis helps us adopt pertinent design principles such as *hardening*, *diversity* and *principle of least privilege* to enhance the resilience of social VRLEs. Through experiments in a vSocial case study, we demonstrate the effectiveness of our attack tree modeling with a reduction of 26.82% in probability of disruption of loss of integrity (security) and 80% in privacy leakage (privacy) in before and after scenarios pertaining to the adoption of the design principles.

Keywords: Virtual reality learning application · attack trees · security · privacy · formal verification.

1 Introduction

Social Virtual Reality (VR) is a new paradigm of collaboration systems that uses edge computing for novel application areas involving virtual reality learning environments (VRLE) for special education, surgical training, and flight simulators. Typical VR system applications comprise of interactions that require coordination of diverse user actions from multiple Internet-of-Things (IoT) devices, processing activity data and projecting visualization to achieve cooperative tasks. However, this flexibility necessitates seamless interactions with IoT devices, geographically distributed users outside the system's safe boundary, which poses serious threats to security and privacy [1].

Although existing works [2], [3] highlight the importance of security and privacy issues in VR applications, there are a limited systematic efforts in evaluating

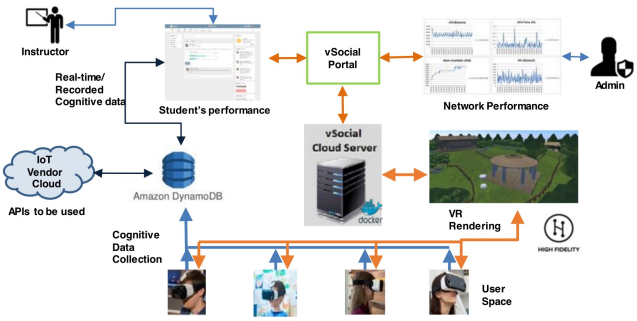


Fig. 1: vSocial system components used for real-time student learning environment

the effect of various threat scenarios on such edge computing based collaborative systems with IoT devices. Specifically, VRLE applications are highly susceptible to Distributed Denial of Service (DDoS) attacks, due to the distributed IoT devices (i.e., VR headsets) connecting to virtual classrooms through custom controlled collaboration settings. Moreover, loss of confidential information is possible as VRLEs track student engagement and other real-time session metadata.

In this paper, we consider a VRLE application designed for youth with autism spectrum disorder (ASD) as case study viz., vSocial[3] [4]. Our multi-modal VRLE system as shown in Figure 1 renders 3D visualizations based on the dynamic human computer interactions with an edge cloud i.e., vSocial Cloud Server. VRLE setup includes: VR headset devices (HTC Vive), hand-held controllers, and base stations for accurate localization and tracking of controllers [4]. Any disruption caused by an attacker with malicious intent on the instructor's VR content or administrator privileges will impact user activities and even cause cybersickness. Failure to address these security and privacy issues may result in alteration of instructional content, compromise of learning outcomes, access privileges leading to confidential student information disclosure and/or poor student engagement in ongoing classroom sessions.

Motivated by the importance of ensuring security and privacy in the VRLE application, we propose a novel framework for quantitative evaluation of security and privacy metrics using attack trees to perform Statistical Model Checking (SMC)[5]. Attack trees are graphical models, which provide a systematic representation of attack scenarios. They are popular but lack support for modeling the temporal dependencies between the attack tree components. We overcome this limitation by utilizing an automated state-of-the-art SMC tool UPPAAL [5]. This involves translating the constructed security and privacy attack tree of the VRLE application into the Stochastic Timed Automata (STA) in a compositional manner. We define *security* as a condition that ensures a VR system to

[3] Moving forward we use the acronym VRLE to refer to our case study application viz., vSocial.

perform critical functions with the establishment of confidentiality, integrity and availability [6]. *Privacy* is defined as a property that regulates the IoT data collection, protection, and secrecy in such interactive systems [6]. The main paper contributions summary is as follows:

- We propose a framework to evaluate security and privacy of VR applications using the attack tree formalism and statistical model checking. To show the effectiveness of our solution approach, we utilize a VRLE application case study viz., vSocial that uses edge computing assisted IoT devices for collaboration.
- We perform a trade-off analysis by evaluating the severity of different types of attacks on the considered VRLE application. We observe that unauthorized access and causes of DoS attack are the most vulnerable candidates.
- We demonstrate the effectiveness of using design principles (also known as security principles) i.e., *hardening, diversity, principle of least privilege* on the privacy and security of VRLE application in the event of most severe threats. We show that in terms of security – a combination of {*hardening, principle of least privilege*} is most influential in reducing disruption of Loss of Integrity (LoI). Similarly for privacy, a combination of {*diversity, principle of least privilege*} is most influential in reducing privacy leakage.

The remainder of the paper is organized as follows: Section 2 discusses related works. Section 3 covers the background of the security and privacy problem involving edge computing and IoT devices. Section 4 discusses the proposed security and privacy framework in detail. Section 5 presents the numerical results using our proposed framework on the VRLE case study. Section 6 discusses the effectiveness of design principles on impactful attacks. Section 7 concludes the paper.

2 Related Works

There have been several comprehensive studies that highlight the importance of security and privacy threats on IoT and related paradigms such as Augmented Reality (AR) with IoT devices, and edge computing. A recent study [1] on these challenges in AR and VR discusses the threat surface area for educational initiatives without characterizing the attack impact. Survey articles [2], [3], [7] are significant IoT related works for understanding the concepts of threat taxonomy and attack surface area. Literature about security, privacy attacks [7], [8], [9] in IoT, fog computing surveys highlight the need to go beyond specific component such as network, hardware or application and suggest a focus on end-to-end solutions that considers the inherent system and data vulnerabilities. An observation given the above state-of-art is that there are very few scholarly works on the quantitative evaluation for these security and privacy issues in the context of VR applications. Attack trees in cyber-physical systems involving SCADA system [10], are one of the preliminary threat modeling approaches to determine the probability of detection and severity of threats. One of our preliminary works [11] shows how to perform risk assessment using security, privacy and

safety metrics of the VRLE applications utilizing an attack tree simulation tool. In contrast, our work focus is on formal modeling of attack trees using STAs and utilizes a state-of-the-art formal verification tool to evaluate the developed security and privacy attack trees. In addition, our proposed framework incorporates the concept of design principles to enhance the security and privacy of VRLE applications.

Amongst the numerous prior works on attack trees, the work in [12] presents a novel concept of Attack Fault Tree (AFT), a combination of both attack and fault trees. A model of STA [13] alleviates some assumptions made in timed automata and provides advantages such as choice of transitions requiring satisfaction of very precise clock constraints. Timed automata [14] provides an abstract model of the real system by using clocks as well as timing constraints on the transition edges. As compared to Continuous-Time Markov Chains (CTMC) [15], STA models allow us to express hard time constraints such as $x \leq t$. We studied the above existing modeling techniques to formalize our security and privacy attack trees into STA, which can be evaluated using model checking tools such as the UPPAAL SMC [16].

3 Preliminaries

3.1 Statistical model checking

Statistical model checking (SMC) is a variation of the well-known classical model checking [17] approach for system that exhibits stochastic behavior. The SMC approach to solve the model checking problem involves simulating the system for finitely many runs, and using hypothesis testing to infer whether the samples provide a statistical evidence for the satisfaction or violation of the specification [18]. In contrast to the classical Monte Carlo simulation and analytical methods, SMC is based on a formal semantic of systems which allows us to reason on their complex behavioral properties through detailed temporal constraints on the system's executions.

Stochastic timed automata: Stochastic timed automata (STA) is an extended version of timed automata (TA) with stochastic semantics. A STA associates logical locations with continuous, generally distributed sojourn times [14]. In STA, constraints on edges and invariants on locations, such as clocks are used to enable transition from one state to another [12].

Definition 1 (Stochastic timed automata). *Given a timed automata which is equipped with assignment of invariants $\mathcal{I}$ to locations $\mathcal{L}$, we formulate an STA as a tuple $T = \langle\ \mathcal{L},\ l_{init},\ \Sigma,\ \mathcal{X},\ \mathcal{E},\ \mathcal{I},\ \mu\ \rangle$, where $\mathcal{L}$ is a finite set of locations, $l_{init} \in \mathcal{L}$ is the initial location, Σ is a finite set of actions, $\mathcal{X}$ is the finite set of clocks, $\mathcal{E} \subseteq \mathcal{L} \times \mathcal{L}_{clk} \times \Sigma \times 2^{\mathcal{X}}$ is a finite set of edges, with $\mathcal{L}_{clk}$ representing the set of clock constraints, $\mathcal{I}$: $\mathcal{L} \longrightarrow \lambda$ is the invariant where λ is the rate of exponential assigned to the locations $\mathcal{L}$, μ is the probability density function (μ_l) at a location $l \in \mathcal{L}$.*

An exemplar STA is shown in Figure 2 that consists of the locations {Initial, Wait, Fail}. Herein, the initial location represents the start of execution of an STA and a *clock_x* is used to keep track of the global time. The communication

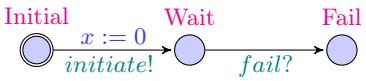


Fig. 2: An exemplar STA

in an STA exists between its components using message broadcast signals in a bottom-up approach. The STA is activated by broadcasting *initiate!* signal, which transitions to wait location and waits for the *fail* signal. Based on the reception of the *fail?* signal, the STA is transitioned from wait location to Fail location, which is the final state that represents the disruption of the system as shown in Figure 2. In an STA, time delays are governed as probability distributions (used as invariants) over the locations. The Network of Stochastic Timed Automata (NSTA) is defined by composing all component automaton to obtain a complete stochastic system satisfying the general compositionality criterion of TA transition rules [5].

Definition 2 (Network of stochastic timed automata). *Given a network of STA $\mathcal{S}_T$ using the parallel composition of STA of all the nodes in an attack tree A, we formulate an NSTA as a model using the transition rules [5] $\mathcal{S}_T = \mathcal{S}_{v_1} // \mathcal{S}_{v_2} //..... // \mathcal{S}_{v_n} // Top\_event$ where // is the parallel composition operator which allows the composition of individual STA of several nodes N of an attack tree A denoted as $\mathcal{S}_{v_1}$,.....,$\mathcal{S}_{v_n}$, and Top_event is the root node of the attack tree A.*

Each of these individual stochastic timed automata $\mathcal{S}_{\mathrm{v_1}}$,.....,$\mathcal{S}_{\mathrm{v_n}}$ are equipped with assignment of invariants $\mathcal{I}$ to locations $\mathcal{L}$, shared action messages $\Sigma = \Sigma_{\mathrm{b}} \cup \Sigma_{\mathrm{r}}$ partitioned into broadcast (Σ_{b}) and receptions Σ_{r}, where broadcasts are of the form !m and receptions of the form ?m

3.2 UPPAAL SMC

In this work, we use the UPPAAL SMC model checker for analyzing the attack trees. The UPPAAL SMC is an integrated tool for modeling, validation and verification of real-time systems modeled as network of stochastic timed automatas (NSTAs) extended with integer variable, invariants and channel synchronizations. The modeling language of UPPAAL offers additional features such as bounded variables which can be used to model the behaviour of the real-time systems. The SMC query language is a weighted extension of the Metric Interval Temporal Logic (MITL) which is used to express over runs [16]. Detailed syntax and semantics of the SMC queries can be found in [19]. In the SMC approach, the number of simulations N to find the probability estimate is automatically derived using an estimation algorithm and statistical parameters, such as $1-\alpha$ (required confidence interval) and ϵ (error bound). Probability estimation for the mentioned query is returned as an output after N number of simulations, with confidence interval $1-\alpha$ and error bound ϵ. Output of the query is given as a estimation interval $[Pr-\epsilon, Pr+\epsilon]$, where Pr is the probability that a random run satisfies φ, where φ is a MITL formula. For instance, if we indicate the

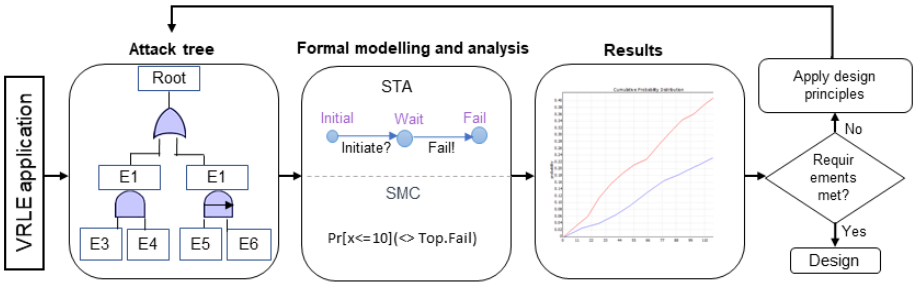


Fig. 3: Proposed framework for security and privacy analysis of social VRLE applications in order to adapt the system design

goal state in the STA of *Top_event* as *Fail*, then the probability of a successful disruption within time t can be written as: Pr[x <= t] (<> Top_event.Fail), where <> represents the existential operator ($\Diamond$) and x is a clock in the STA to track the global time.

3.3 Design principles

To build a trustworthy system architecture which ensures security and privacy, integration of design principles in the life cycle of edge computing interconnected and distributed IoT device based systems is essential [20]. We define and use these design principles adapted from NIST SP800-160 [6] and [20]: *Hardening* – is defined as reinforcement of individual or types of components to ensure that they are harder to compromise or impair. *Diversity* – is defined as implementation of a feature with diverse types of components to restrict the threat impact from proliferating further into the system. *Principle of least privilege* – is defined as limiting the privileges of any entity, that is just enough to perform its functionality and prevents the effect of threat from propagating beyond the effected component.

4 Proposed framework

In this section, we present a framework that translates the security and privacy attack trees into an equivalent STA representation, in order to evaluate the threats and vulnerabilities on the underlying VRLE system [4]. We analyze the threats using the STAs to form an NSTA which in turn is fed to the statistical model checking tool UPPAAL SMC, as shown in Figure 3. With this quantitative assessment, we determine if the probability of disruption is higher than a set threshold, and subsequently adopt design principles such as: *hardening*, *diversity* and *principle of least privilege* to the generated attack trees. Overall, our framework investigates potential design alternatives based on design principles to recommend the best candidate for securing a edge computing based VRLE with a known attacker profile.

4.1 Attack trees

Attack trees are hierarchical models that show how an attacker goal (root node) can be refined into smaller sub-goals (child/intermediate nodes) via gates until no further refinement is possible such that the basic attack steps (BAS) are reached. BAS represents the *leaf nodes* of an attack tree [21]. The *leaf nodes* and the *gates* connected in the attack trees are termed as *attack tree elements*. To explore dependencies in attack surfaces, attack trees enable sharing of subtrees. Hence, attack trees are often considered as directed acyclic graphs, rather than trees [12].

Definition 3 (Attack trees). *An attack tree A is defined as a tuple {N,Child, Top_event, l} ∪ {AT_elements} where, N is a finite set of nodes in the attack tree; Child: $N \rightarrow N^*$ maps each set of nodes to its child nodes; Top_event is an unique goal node of the attacker where Top_event ∈ N; l: is a set of labels for each node $n \in N$; and AT_elements: is a set of elements in an attack tree A.*

Attack tree elements: Attack tree elements aid in generating an attack tree and are defined as a set of $\{G \cup L\}$ where, G represents gates; L represents leaf nodes. Following are the descriptions of each of the AT elements.
Attack tree gates: Given an attack tree A, we formally define the attack tree gates $G = \{OR, AND, SAND\}$[4]. With the capability of a multi-step threat scenario in attack trees, each level in the attack tree can be modeled using a composition of gates such as AND, OR and Sequential AND (SAND) gates. An AND gate is disrupted when all its child nodes are disrupted, whereas an OR gate is disrupted if either of its child nodes are disrupted. Similarly, SAND gate is disrupted when all its child nodes are disrupted from left to right using the condition that the success of previous step determines the success of the upcoming child node. A SAND gate is represented with the OR gate symbol with a right arrow appearing inside the gate. Using these gates, we define the *intermediate nodes* of an attack tree. The output nodes of the gates G in an attack tree A are defined as *Intermediate nodes* (I), which will be located at a level that is greater than the leaf nodes.
Attack tree leaves: An attack tree *leaf node* is the terminal node with no other child node(s). It can be associated with *basic attack steps (BAS)*, which collectively represent all the individual atomic steps within a composite attack scenario. To elucidate, for an attacker to perform intrusion as mentioned in Figure 5, the BAS involves: (i) identity spoofing, and (ii) unauthorized access to the system. To perform an intrusion, the BAS includes: unauthorized login via SQL injection or identity spoofing, which are dependent on the attacker profile. Thus, every BAS appears as a leaf of the attack tree and the attack duration is assumed to have an exponential rate such as $P(t) = 1 - e^{\lambda t}$ where, λ is the rate of exponential distribution. We use the exponential distribution because of its tractability and ease of handling, since they are defined by a single parameter.

[4] We restrict ourselves to modeling of these three gates, however attack trees can adopt any other gates from the standard fault trees.

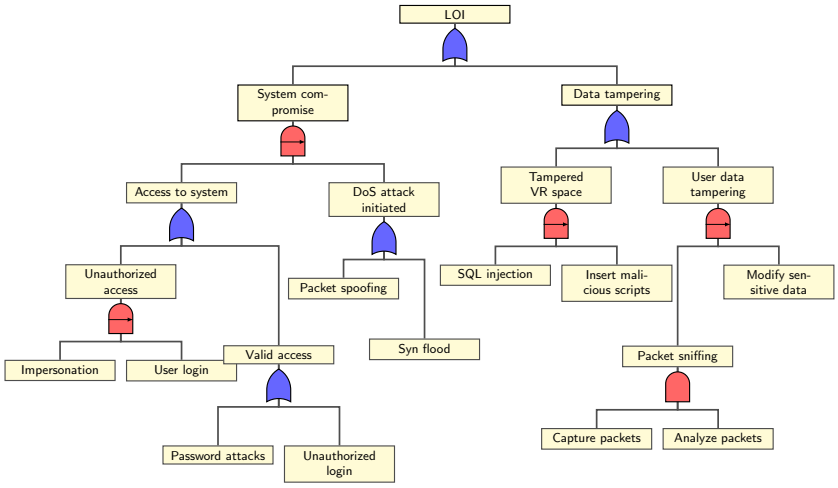


Fig. 4: Attack tree for security factors of VRLE case study viz., vSocial

Attacker profile: An *attacker profile (AP)* denotes the ability of various threat agents to influence the overall system disruption for a given set of resources. It is a single composite term to represent the logical and real value attributes such as budget, resources, duration, along with the *{low, medium, high}* level of the attacker's capability to successfully carry out an attack. For this paper, we adapt the attacker profiles listed in a prior work [21] to simulate a combination of non-deterministic attack steps that disrupt the overall VRLE system.

4.2 Security and privacy attack trees

Herein, we present a discussion of the attack steps illustrated in the security and privacy attack trees shown in Figures 4 and 5. The descriptions of the leaf nodes are listed in Table 1. The first step in the proposed framework is the attack tree formalism based on the respective security and privacy vulnerable surface area in the underlying VRLE system shown in Figure 1. Although the traditional approach of using the CIA triad of {Confidentiality, Integrity, Availability} may initially appear as intuitive, it translates to enormous number of levels and leaf nodes in the tree. Hence, we focus on the security threats pertaining to LoI and refer to any scenarios where system/user data or system components are compromised by an unauthorized entity. On the other hand, we have limited the privacy issues to any scenario leading to a privacy leakage for the VRLE system.

A summary of the LoI scenarios that can either compromise the system security due the loss of data or system components are discussed in the security attack tree shown in Figure 4. An attacker can get unauthorized access to tamper any confidential information by impersonating a valid user. In addition, attacks such as spoofing can allow an attacker to get admin access and extract network and user information. Failure to address such issues can result in LoI of the VRLE system. Similarly, issues related to the other two facets in the CIA triad

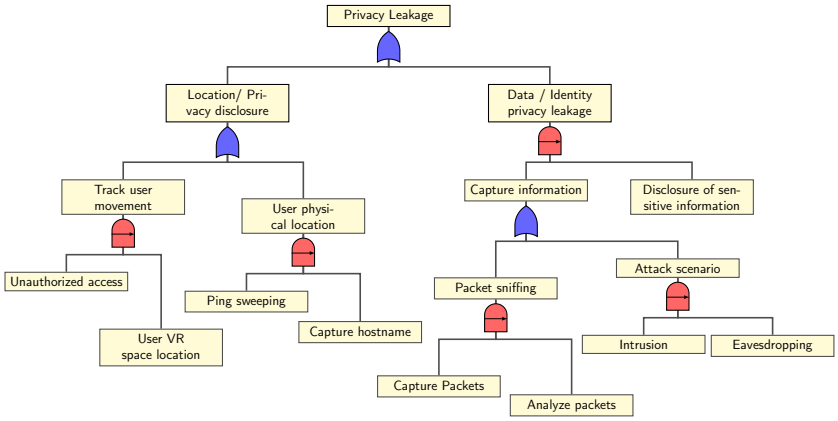


Fig. 5: Attack tree for privacy factors of VRLE case study viz., vSocial

such as Confidentiality and Availability (LoC and LoA) in the VRLE application can be addressed by generating new security attack trees.

On the other hand, the privacy attack tree is built with the threat scenarios causing privacy leakage as shown in Figure 5. An intruder entering the VR world with fake credentials to snoop into the virtual classroom conversation is the most obvious case of user privacy breach. In another scenario, the attacker can capture key confidential information such as login credentials, avatar or even student classroom engagement data, through eavesdropping and can disclose this information to external parties with malicious intents. Although we have presented several privacy-related threats, there can also be an interrelationship between the two attack trees. In other words, temporal data dependencies from security threats can lead to disclosure of sensitive information. Similarly, availability of VR content or data on the instructor/admin web application can be compromised if an attacker performs DoS which can lead to both privacy and security issues. In summary, the trees we presented cover the individual threat scenarios which independently impact security or privacy. However, complex attack trees that consider their inter-dependencies can more realistically expresses the wider attack surface area. Due to space constraints, we have presented efforts only on the individual leaf nodes, and our approach can be easily extended to subtrees from different attack trees.

4.3 Translation of attack trees into stochastic timed automata

In this section, we generate stochastic timed automata (STA) from the corresponding security and privacy attack trees detailed in Section 4.2. An overview of our translational approach is shown in the Figure 6 which includes: (i) each of the leaf nodes in these attack trees are converted into individual STA. The intermediate events, which are basically the output of the logic gates used at different levels are converted iteratively into stochastic timed automata (STA);

Table 1: Descriptions of leaf nodes in security and privacy AT's in the VRLE application case study

Security attack tree		**Privacy attack tree**	
Leaf node components	**Description of leaf nodes**	**Leaf node components**	**Description of leaf nodes**
Impersonation	Attacker successfully assumes the identity of a valid user	Unauthorized Access	Attacker gains access to VR space
Packet Spoofing	Spoofing packets from a fake IP address to impersonate	User VR space location	Attacker determines the user location in VR space
Sync Flood	Sends sync req. to a target and direct server resources away from legitimate traffic	Ping sweeping	Attacker sends pings to a range of IP addresses and identify active hosts
SQL Injection	Attacker injects malicious commands in user i/p query using GET and POST	Capture packets	Attacker uses packet sniffer to capture packet information
Insert Malicious Scripts	Attacker successfully adds malicious scripts in VR	Analyze packets	To identify erroneous packets to tamper
Capture Packets	Attacker uses packet sniffer to capture packet information	Intrusion	Attacker performs an unauthor -ized activity on VR space
Analyze Packets	To identify erroneous packets to tamper	Eavesdropping	Attacker listens to conversation in VR space
Modify sensitive data	To modify any sensitive information by eavesdropping	Disclosure of sensitive information	Attacker releases any captured sensitive data
User login	User login into VRLE	Capture hostname	With IP address obtained, attacker can capture the hostname in VRLE application
Unauthorized login	Attacker gains access into VRLE by unauthorized means		
Password attacks	Attacker recovers password of a valid user		

(ii) the generated STA are composed in parallel by including the root node as defined in (Section 2) to form an NSTA; (iii) the obtained NSTA is used for model checking in order to verify the security and privacy metrics formalized as SMC queries.

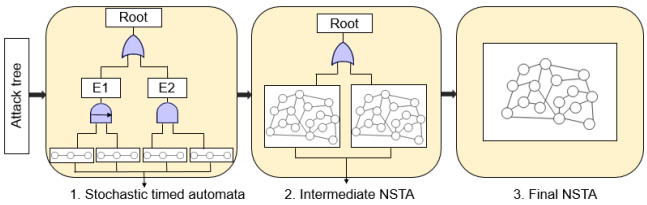
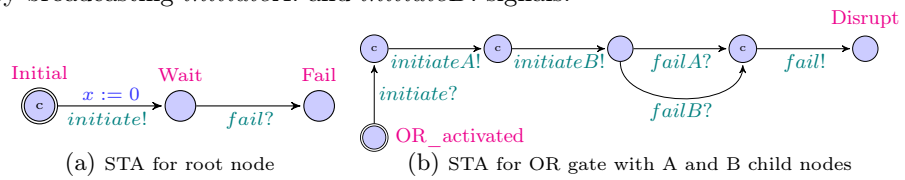


Fig. 6: Logical transformation of compositional aggregation of attack tree models

To demonstrate the translation of an attack tree into an STA, we will consider the security attack tree as shown in Figure 4. As part of the translation, each of the security AT element node (leaf and gates) input signals are connected to the output signal of child nodes. The generated network of STAs communicate using signals such as: *initiate* - indicates activation of attack tree element. This signal is sent at system startup from the top parent node to its children. *fail* - control signal that indicates failure (or disruption) of attack tree element. This signal is sent to parent node from its child node to indicate an STA disruption. The scope of the above signals can also be extended by special symbols. The signals equipped with '?' (ex: initiate?) means that the event will wait for the

reception of the intended signal. Another symbol '!' (ex: initiate!) implies that the output signal is broadcasted to other STAs in the attack trees.

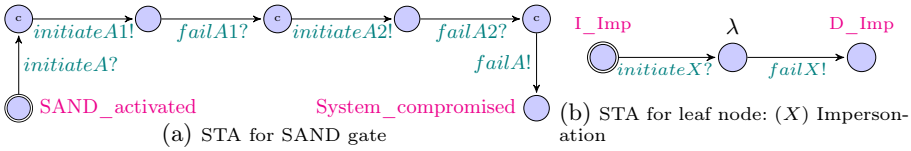
Illustrative example: In the subsequent paragraph, we have explained the above concepts by converting the security attack tree into an exemplar NSTA. Figure 7a shows the converted STA of the LoI i.e., root node (Top_event) equipped with $initiate!$ and $fail?$ signals for the security attack tree. The Top_event is the OR gate output for the two child nodes: (A) "System Compromise" and (B) "Data Tampering". Top OR gate sends an initiate signal and activates its child nodes "System compromise" and "Data Tampering" as shown in Figure 7b by broadcasting $initiateA!$ and $initiateB!$ signals.



(a) STA for root node (b) STA for OR gate with A and B child nodes

Fig. 7: STA for OR gate and root node of security attack tree.

After initialization, if either of the nodes (A) OR (B) are disrupted, then a $fail!$ signal is sent to the Top_event, which forces it to transition from $Wait$ to $Disrupt$ state, representing LoI in the system. We declare clock x, which is a UPPAAL global variable to keep track of the time progression.



(a) STA for SAND gate (b) STA for leaf node: (X) Impersonation

Fig. 8: STA of a) SAND gate b) leaf node for security attack tree

After discussing the disruption of the Top_event in Figure 4, we will now explain the STAs that cause the disruption of its child nodes. The node (A) "System compromise" is disrupted, when it receives a $fail$ signal from its child nodes (A1) "Access to System" AND (A2) "DoS Attack initiated" in a sequential manner as shown in Figure 8a. The node A being a SAND gate, initially activates the leftmost child (A1) and waits for its $fail$ signal to trigger as shown in its equivalent STA Figure 8a. After receiving the $fail$ signal from (A1), it activates (A2). On receiving $fail$ signal from A2, the parent node i.e., (A) "System compromise" gets disrupted. Similarly, the other OR, SAND and AND gates in Figure 4 get activated from the multiple parent nodes due to the component automation of the attack trees for communication using the broadcast signals.

In the following, we discuss the process of generating the STAs of the leaf nodes in the security attack tree. For instance, the STA for the leaf nodes "Impersonation" is shown in Figure 8b. The STA for leaf nodes such as "Login" can be generated by utilizing the same approach. STA for "Impersonation" gets activated after receiving $initiateX!$ signal from the parent, and (λ) value for the

rate of exponential distribution is given on the intermediate node. The parent node disrupts gradually accordingly, once it receives the *failX!* broadcast signal. Based on the λ given at the leaf node, the probability estimation of disruption is calculated, which propagates upwards in the tree to finally calculate the probability of disruption of the root event (LoI). Coupling all the generated STAs, a network will be formed known as NSTA for the security attack. tree as shown in Figure 6. Using similar strategies mentioned above, NSTA for privacy leakage attack tree can be developed. Our proposed framework performs model checking of the VRLE system using generated NSTAs for the security and privacy attack trees with a quantitative analysis that is detailed in Section 6.

5 Numerical results

In this section, we present the results obtained using our proposed framework that is relevant to edge-computing assisted VRLEs that use IoT devices for collaborations. As mentioned in Section 4.2, the threat scenarios we consider include LoI and privacy leakage for security and privacy, respectively. In the following analysis, we assume that our design requirement is to keep the probability of disruption of LoI and privacy leakage below the threshold of 0.25. For evaluation purposes, we use λ as a parametric input to the leaf nodes of the attack trees as shown in Figures 4 and 5 as explained in Section 4.1. The value for λ considered in relation to the different threat scenarios are given in Table 2. These are further applied to calculate the probability estimation (Pr) as mentioned in Section 4.1. We have used the λ values from [22] and [23] with arbitrary numbers. Even though we use arbitrary values, the designed NSTAs can be verified using user specification, and can be changed based on the particulars of the application setting.

Table 2: Rate of exponential values given to the leaf nodes of security and privacy attack trees

Security		**Privacy**	
Security Threats	**Rate Of Exponential**	**Privacy Threats**	**Rate Of Exponential**
Impersonation	0.006892	Unauthorized access	0.006478
User login	0.0089	User VR space location	0.0094
Password attacks	0.008687	Capture hostname	0.002162
Unauthorized login	0.008687	Ping sweeping	0.004162
Packet spoofing	0.0068	Capture packets	0.00098
SYNC flood	0.0068	Analyze packets	0.0048
SQL injection	0.00231788	Disclosure of sensitive info	0.0009298
Insert malicious scripts	0.008	Intrusion	0.006628
Capture packets	0.000 98	Eavesdropping	0.08
Analyze packets	0.0048	–	–
Modify sensitive data	0.002642	–	–

The process used for the LoI case can be extended to include the other CIA triad components such as Loss of Confidentiality and Loss of Availability in a similar manner, but we do not include them intentionally to keep the scope of the design space manageable. After providing λ value as a parametric input to the

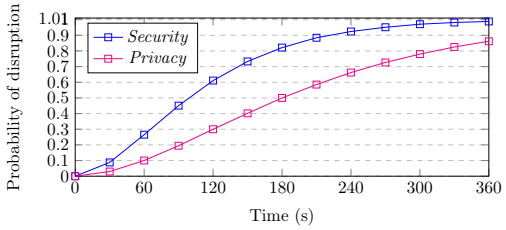


Fig. 9: Probability of disruption of LoI and privacy leakage w.r.t time window for attacker

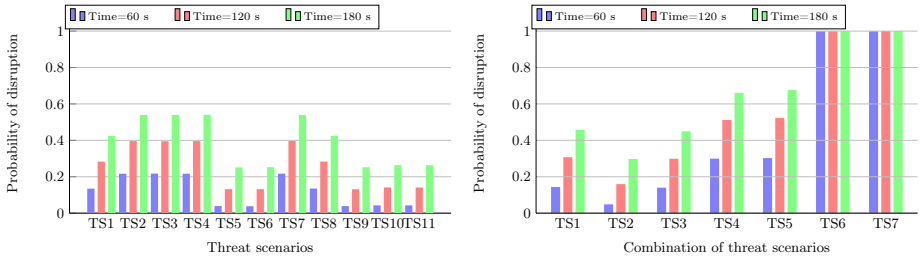
leaf nodes, UPPAAL SMC queries as explained in Section 3.2 can be utilized over the generated NSTA to find the disruption probability of the respective attack trees. In addition to this, it is important to consider the error bound for this type of quantitative analysis. UPPAAL tool gives 95% confidence interval for the probability of disruption with an error bound of ϵ. Better accuracy for the probability of the Top_event can be achieved using a lesser error bound, but at the expense of time required to run the queries. However, as a trade-off - when we reduce the error bound to very low values, the time required to calculate the probability also increases depending on the model complexity. Therefore, we use a value of 0.001 for ϵ in our experiments. In summary, we consider root nodes i.e., LoI (security attack tree) and privacy leakage (privacy attack tree) as goal nodes and provide a quantitative analysis of potential vulnerabilities in the VRLE system.

5.1 Evaluation of security and privacy threats

Using the constructed STA for security and privacy attack trees discussed in Section 4.3, we study the effect on probability of disruption of goal nodes with respect to attacker time window as shown in Figure 9. The characteristic results shows that - for the assumed values of λ, the probability of disruption for goal nodes increases more than the considered requirement threshold value which is 0.2. Thus, we quantify the threats affecting goal nodes in security and privacy attack trees by performing analysis on: (i) individual, and (ii) combination of leaf nodes.

Vulnerability Analysis in security attack trees As part of the numerical analysis, we iteratively assign a maximum value of λ as 1 to a leaf node, while maintaining the others at a very small positive constant $(K) \approx 0.002$. The following discussion explains the numerical analysis of: (i) individual leaf nodes, and (ii) combination of leaf nodes in the security attack trees, to determine their effect on the probability of disruption in the LoI case.

Leaf node analysis in security attack tree: As part of the study of individual leaf nodes effect on LoI, we analyze the probability of disruption over multiple time windows for each leaf node in the security attack tree as shown in the clustered column bar graphical Figure 10a. We perform a thorough analysis of this smaller set of leaf nodes by simulating these values in 30 second



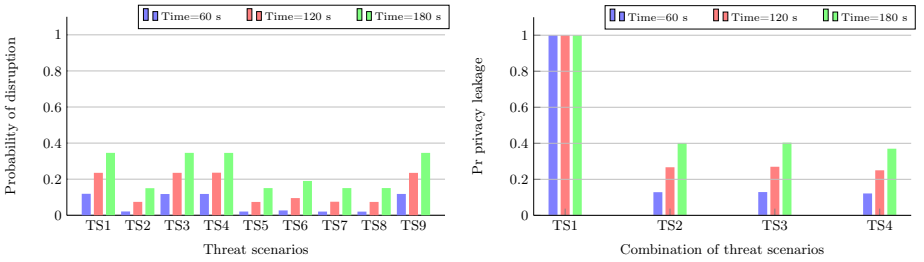
(a) TS in security AT at $t \leq 180$ s where - TS2, TS3, TS4, TS7 are the most vulnerable nodes.

(b) TS in security AT at t $\leq$ 180 seconds where – TS6, TS7 are the most vulnerable combination

Fig. 10: Vulnerability analysis a)individual b)combination of leaf nodes in security AT

time intervals i.e., t = 0, 30, 60,.., 180. We term all the threat scenarios (TS) for individual leaf node analysis in security attack tree as: $TS1$–insert malicious scripts, $TS2$– packet spoofing, $TS3$– unauthorized login, $TS4$– password attacks, $TS5$– modify data, $TS6$– analyze packets, $TS7$– Sync flood, $TS8$– SQL injection, $TS9$– capture packets, $TS10$– impersonation, $TS11$– user login as shown in Figure 10a. Using these terms, we deduce that the high probability values of 0.53 at $t \leq 180$, the leaf nodes $TS3$: *unauthorized login*, $TS4$: *password attacks* (for an authorized access) and $TS2$: *packet spoofing*, $TS7$: *sync flood* (DoS attack) as the most vulnerable components in the security attack tree as listed in Table 3. We use this smaller subset of leaf nodes for deeper analysis. We begin by investigating the combination of multiple such nodes on the disruption probability of the goal nodes for the security attack tree. Furthermore, our in-depth examination on these specific vulnerable leaf nodes of the security attack tree helps us to identify how they impact LoI in conjunction with other highly vulnerable nodes.

Vulnerability analysis for combination of leaf nodes in security attack tree: In this section, we investigate the combination of several leaf nodes which either belong to the same or different sub-trees of the security attack tree. We determine the most vulnerable combinations of leaf nodes that affect the probability of disruption of LoI. In this analysis, we ignore other less-impacting individual leaf nodes for space constraints. We are mainly using two types of leaf node combinations. Firstly, those that influence the same intermediate node of the security attack tree. For example, {"Impersonation", "Packet spoofing"} both effect "System compromise" as shown in Figure 4. Secondly, leaf nodes belonging to two different sub-trees: {"System compromise", "Data tampering"}. An example for this type of combination is the {"SQL injection", "Password attacks"}. Similarly, we perform the numerical analysis outlined in analysis of individual_leaf nodes of security AT, but here combination of threat scenarios (TS) are considered. We enlist the combination of threat scenarios as $TS1$ – {impersonation,SQL injection}, $TS2$ – {impersonation, modify data}, $TS3$ – {(SQL injection, capture packets}, $TS4$ – {(pwd attacks, SQL injection}, $TS5$ – {impersonation, packet spoofing}, $TS6$ – {packet spoofing, unauth login}, $TS7$ – {unauthorized login, sync flood}. A summary of the findings are presented in Figure 10b, where we deduce that $TS6$ – {"Packet spoofing", "Unauthorized



(a) TS in privacy AT at $t \leq 180$ sec where - TS1, TS3, TS4, TS9 are the most vulnerable nodes.

(b) TS in privacy AT at t ≤ 180 seconds where – TS1 is the most vulnerable combination

Fig. 11: Vulnerability analysis a)individual b)combination of leaf nodes in privacy AT

login"}, $TS7$ – {"Unauthorized login", "Sync flood"}, as the most vulnerable combination of threat scnearios with probability of disruption on LoI at $t \leq 180 = 1$ as listed in Table 3. This will qualify the resultant vulnerable combination of leaf nodes for further mitigation measures according to the design principles. In order to understand the impact of one of the most vulnerable combination $TS6$ – {"Packet Spoofing", "Unauthorized login"}, we performed a detailed numerical analysis where the probability of disruption reaches to maximum at $t = 9$s and remains constant after $t > 9s$, which marks the time taken to disrupt the system for our assumed attacker profile.

Vulnerability Analysis in privacy attack trees In this section, we analyze the privacy attack tree using a similar approach discussed in Section 5.1 at: (a) individual leaf nodes, and (b) combination of leaf nodes. For the individual leaf node analysis of privacy attack tree shown in Figure 5, we term these nodes as $TS1$ – unauthorized access, $TS2$ – capture packets, $TS3$ – user VR space location, $TS4$ – ping sweeping, $TS5$ – analyze packets, $TS6$ – disclosure of sensitive info, $TS7$ – intrusion, $TS8$ – eavesdropping, $TS9$ – capture hostname. From the analysis shown in Figure 11a, we deduce that - at $t \leq 180$, the most vulnerable leaf nodes are: $TS1$ – *Unauthorized access*, $TS3$ – *User VR space location*, $TS4$ – *Ping sweeping*, $TS9$ – *Capture hostname* with probability of disruption of privacy leakage is 0.34 as listed in Table 3. Similarly, the analysis on combination of leaf nodes in the privacy attack tree is performed using the approach presented in the vulnerability analysis of the security attack tree. Therefore, we term the combination of threat scenarios (TS) as: $TS1$ – {unauthorized access, user VR space location}, $TS2$– {capture packets, disclosure of sensitive information}, $TS3$ – {unauthorized access, disclosure of sensitive information}, $TS4$ – {capture packets, analyze packets} as shown in Figure 11b. Based on this, we determine that $TS1$ – {"Unauthorized access", "User VR space location"} is the most vulnerable combination of threats for privacy leakage with consistent probability of disruption of 100% through the entire analysis period as listed in Table 3. To study the change in probability of disruption in privacy, we analyze one of the most vulnerable threat scenario combinations TS1{Unauthorized Ac-

Table 3: Most vulnerable components considering the individual & combination of leaf nodes

Level in attack trees	Analysis on security AT		Analysis on privacy AT	
	Different Scenarios	**Identified vulnerable components in security AT**	**Different Scenarios**	**Identified vulnerable components in privacy AT**
Individual leaf nodes	Leaf nodes where probability of disruption in LoI at (tâĽď180) = 0.53	Unauthorized login	Leaf nodes where probability of disruption in privacy leakage at (tâĽď180) = 0.34	Unauthorized access
		Packet spoofing		User location in VR
		Sync flood		Ping sweeping
		Password attacks		Capture hostname
Combinat -ion of leaf nodes	Leaf nodes where probability of disruption in LoI at (tâĽď180) = 1	{Unauthorized login, Packet spoofing}, {Unauthorized login, Sync flood}	Leaf nodes where probability of disruption in privacy leakage at (tâĽď180) = 1	{Unauthorized access, user location}

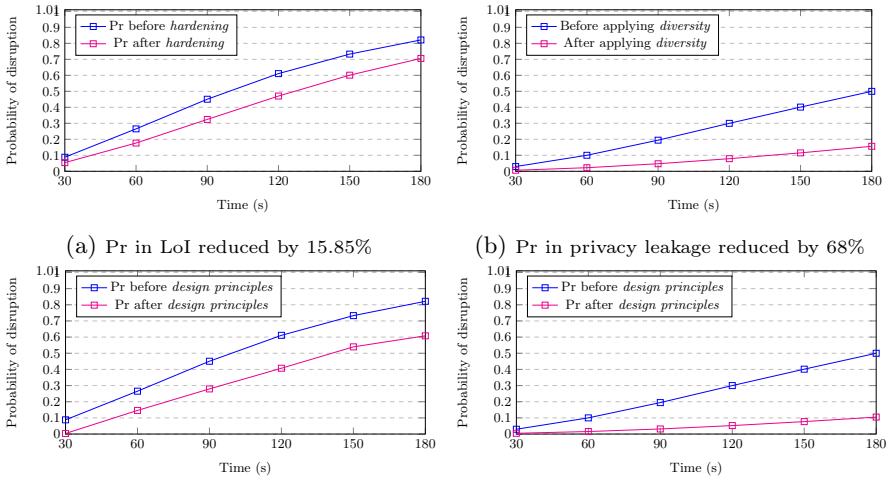
cess, User VR space location}. We find that the disruption probability of privacy leakage reaches to maximum after t $\geq$ 10s giving indications about the minimum time required for the corresponding disruption.

In summary, with such a numerical analysis of attack tree components we can conclude that LoI and privacy leakage concerns needs to be addressed as a part of mitigation measures, and it is feasible to identify design principle candidates to ensure a trustworthy VRLE system.

6 Recommended design principles

In this section, we are examining the effect of applying various design principles to the most vulnerable components identified in the Sections 5.1, and 5.1. Existing works such as NIST SP800-160 document [6], [20] suggest that the services for safeguarding security and privacy are critical for successful operation of current devices and sensors connected to physical networks as part of IoT systems. As mentioned in Section 3.3, these design principles are essential to construct a trustworthy edge computing based system architecture. The goal is to apply a combination of design principles at different levels of abstraction to help in developing effective mitigation strategies. We adopt a selection of design principles such as *hardening*, *diversity* and *principle of least privilege* among the list of principles available in NIST document [6], and [20]. In the following, we demonstrate their effectiveness by showing that there is a reduction in the probability of disruption terms after adopting them in our VRLE system design.

Implementation of design principles on security attack tree: In this section, we apply design principles on one of the identified vulnerable nodes of the security attack tree as shown in Section 5.1. For instance, we incorporate *hardening* design principle on the password attacks, to study its effects on the security metric LoI as shown in Figure 12a. As part of the *hardening* principle, we added new nodes such as firewall and security protocol in the security attack tree. Our results show that the probability of disruption of LoI is reduced from 0.82 to 0.69 (15.85%), with the given attacker profile. The decrease in the disruption of LoI is due to the rise in additional resources that are required by the attacker to compromise such a VRLE application system which is incorporating the *hardening* principle. Similarly, we apply the *principle of least privilege* on



(a) Pr in LoI reduced by 15.85%

(b) Pr in privacy leakage reduced by 68%

(c) Pr in LoI is reduced by 26.82%.

(d) Pr in privacy leakage reduced by 80%

Fig. 12: Effect on security and privacy AT due to application of design principles

the security attack tree, which reduced the probability of disruption of LoI from 0.82 to 0.79 (3.66%). In a similar manner, we applied the design principles to the other vulnerable system components and studied the comprehensive effect on LoI for these different scenarios.

Implementation of design principles on privacy attack tree: Using the similar approach mentioned in design principles on security attack tree, we apply *diversity* design principle on one of the identified vulnerable nodes (unauthorized access) in the privacy attack tree. After adding multiple authentication procedures as part of the *diversity* principle, the probability of disruption on privacy leakage is reduced significantly from 0.5 to 0.16 (68%) as shown in Figure 12b. Similarly, we apply the *principle of least privilege* on the privacy attack tree where the probability of disruption of privacy leakage is slightly reduced from 0.5 to 0.48(4%). Thus, from the above implementation of individual design principles, we conclude that *hardening* and *diversity* are more effective in reducing the disruption of LoI and privacy leakage, respectively. Although, the *principle of least privilege* was incorporated in both the trees, there was no significant effect observed. Thus, our findings emphasizes the benefits in implementing a combination of design principles in both security and privacy attack trees to improve the attack mitigation efforts.

To study the effect on disruption of the LoI and privacy leakage, we adopt a combination of design principles such as: (i) for the security attack tree: {*hardening*, *principle of least privilege*}, and (ii) for the privacy attack tree: {*diversity*, *principle of least privilege*}. We observe that there is a significant drop in the probability of disruption of LoI from 0.81 to 0.6, and 0.5 to 0.1 for privacy leakage as shown in Figures 12c and 12d, respectively.

From the above numerical analysis, we can conclude that incorporating relevant combination of standardized design principles and their implementation

have the potential to better mitigate the impact of sophisticated and well-orchestrated cyber attacks on edge computing assisted VRLE systems with IoT devices. In addition, our above results provide insights on how the adoption of the design principles can provide the necessary evidence to support a trustworthy level of security and privacy for the users in VRLE systems that are used for important societal applications such as: special education, surgical training, and flight simulators.

7 Conclusion

Virtual Reality Learning Environment applications are a new form of collaborative immersive systems that use edge computing and IoT devices, where security and privacy is of critical importance. In this paper, we presented a novel framework that explores new attack surfaces using a prominent formalism known as "attack trees" to investigate the impacts in a VRLE case study viz., vSocial. We developed the security and privacy attack trees for a special education VRLE application and formalized the impact using the stochastic timed automata (STA). The developed STAs were evaluated using the statistical model checking tool UPPAAL to assess the VRLE's security and privacy metrics. Using this approach, we identified the most vulnerable leaf nodes and the combination of leaf nodes that could potentially cause major disruption in LoI and privacy leakage to the application users working together across geographically distributed locations. Furthermore, we illustrated the effectiveness of our framework by comparing the probability of disruption using a combination of 'before' and 'after' scenarios involving the integration of candidate design principles. An interesting observation from our experiments pertains to our results that show that *{hardening, principle of least privilege}* is the best design principle to adopt for enhancing the security, whereas *{diversity, principle of least privilege}* suits better for improving the privacy of the VRLE application.

As part of our future work, we will extend the modeling of attack trees that are originally inspired by fault trees to study the effects of faults and attacks together [12] along with software and hardware maintenance on VRLE applications as pioneered in [24]. This will further allow us to analyze design trade-offs by considering another key concern of 'safety' (that impacts the well-being of the users) in VRLE sessions that are edge computing assisted, and use IoT devices.

References

1. B. Fineman, N. Lewis, "Securing Your Reality: Addressing Security and Privacy in Virtual and Augmented Reality Applications", *EDUCAUSE Review*, 2018. [Online]. Available: https://er.educause.edu/articles/2018/5/securing-your-reality-addressing-security-and-privacy-in-virtual-and-augmented-reality-applications.
2. W. Zhou, Y. Jia, A. Peng, Y. Zhang, P. Liu, "The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved", *IEEE Internet of Things Journal*, 2018.

3. K. Fu, T. Kohno, D. Lopresti, E. Mynatt, K. Nahrstedt, S. Patel, D. Richardson, B. Zorn, "Safety, Security, and Privacy Threats Posed by Accelerating Trends in the Internet of Things", *Computing Community Technical Report*, 2017.
4. C. Zizza, A. Starr, D. Hudson, S. S. Nuguri, P. Calyam and Z. He, "Towards a Social Virtual Reality Learning Environment in High Fidelity", *15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2018.
5. A. David, K. G. Larsen, A. Legay, M. MikuËĞcionis, and D. B. Poulsen, "Uppaal SMC Tutorial", *Int. J. on Software Tools for Tech. Transfer*,2015
6. "Systems Security Engineering:Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems", 2016, [Online]. Available: `https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-160.pdf`
7. R. Roman, J. Lopez, M. Mambo, "Mobile Edge Computing, Fog etal.: A Survey and Analysis of Security Threats and Challenges", *Elsevier FGCS*, 2016.
8. S.Yi, Z.Qin, Q.Li, "Security and Privacy Issues of Fog Computing: A Survey", *Proc. Intl. Conf. Wireless Algorithms, Systems, Applications*, 2015.
9. M. A. Khan, K. Salah, "IoT security: Review, Blockchain Solutions, and Open Challenges", *Elsevier Future Generation Computer Systems*, 2018.
10. E. Byres, M. Franz, D. Miller, "The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems", *IEEE Conf. International Infrastructure Survivability Workshop*, 2006.
11. A. Gulhane, A. Vyas, R. Mitra, R. Oruche, G. Hoefer, S. Valluripally, P. Calyam, K. A. Hoque, "Security, Privacy and Safety Risk Assessment for Virtual Reality Learning Environment Applications", *16th IEEE Annual Consumer Communications & Networking Conference (CCNC) (To Appear)*, 2019.
12. R. Kumar, M. Stoelinga, "Quantitative Security and Safety Analysis with Attack-Fault Trees", *IEEE 18th Int. Symposium on HASE*, 2017.
13. N. Bertrand, P. Bouyer, T. Brihaye, Q. Menet, C. Baier, M. Grosser, M. Jurdzinski, "Stochastic Timed Automata", *Logical Methods in Comp. Sci.*, 2014.
14. P. Ballarini, N. Bertrand, A. Horvath, "Transient Analysis of Networks of Stochastic Timed Automata Using Stochastic state classes", *Springer*, 2013.
15. A. Aziz, K. Sanwal, V. Singhal, R. Brayton, " Model-checking Continuous-time Markov chains", *ACM Transactions on Computational Logic*, 2000.
16. D. Alexandre, K. Larsen, A. Legay, M.Mikučionis ,D. Poulsen, "Uppaal SMC Tutorial", *Int.J. on Software Tools for Technology Transfer*, 2015.
17. E. M. Calrk jr, O. Grumberg, D. Peleg, "Model checking", *MIT Press*, 2000.
18. H. L. S. Younes, M. Kwiatkowska, G. Norman, D. Parker, "Numerical vs Statistical Probabilistic Model Checking", *Int. J. on Software Tools for Technology Transfer*, 2006.
19. P. Bulychev, A. David, K.G. Larsen, A. Legay, G. Li, D. B. Poulsen, "Rewrite-based Statistical Model Checking of wmtl", *Int. Conference on Runtime Verification*, 2012.
20. A. Laszka, W. Abbas, Y. Vorobeychik, X. Koutsoukos,"Synergistic Security for the Industrial Internet of Things: Integrating Redundancy, Diversity, Hardening", *IEEE ICII*, 2018.
21. G. Norman, D. Parker, J. Sproston, "Model checking for probabilistic timed automata", *Formal Methods in System Design*, Vol. 17, pp. 164–190, 2013.
22. P. Saripalli, B. Walters, "QUIRC: A Quantitative Impact and Risk Assessment Framework for Cloud Security", *IEEE 3rd Int. on Cloud Computing*, 2010.
23. M. Kiani, A. Clark, and G. Mohay, "Evaluation of anomaly based character distribution models in the detection of SQL injection attacks", *3rd Int. Conference on Availability, Reliability and Security*, pp. 47-55, 2008.

24. N. Cauchi, K. A. Hoque, A. Abate, M. Stoelinga, “Efficient Probabilistic Model Checking of Smart Building Maintenance using Fault Maintenance Trees", *Proc. of 4th ACM Int. Conference on Systems for Energy-Efficient Built Environments*, 2017.