Attack Trees for 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 formalism and model checking techniques in order 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 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 · Special Education · Security · Privacy · Attack trees · 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 the effect of various threat scenarios on such edge computing based collaborative

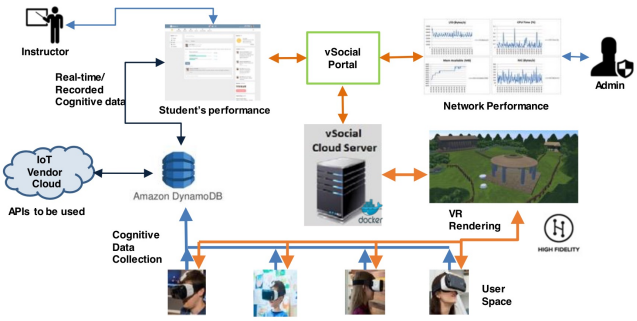


Fig. 1: vSocial system components used for real-time student learning environment

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 [6]. 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 perform critical functions with the establishment of confidentiality, integrity and

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

availability [7]. *Privacy* is defined as a property that regulates the IoT data collection, protection, and secrecy in such interactive systems [7]. 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. From our results, we observe that: i) unauthorized access and causes of DoS attack (in security attack tree), ii) track user movement and user physical location (in privacy attack tree) are the most vulnerable candidates in VRLE.
- 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 the probability 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 discusses the background essential for formulating the VRLE security and privacy problem. 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 threat scenarios. 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], [8] are significant IoT related works for understanding the concepts of threat taxonomy and attack surface area. Literature about security, privacy attacks [8], [9], [10] 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 [11], are one of the preliminary threat modeling approaches to determine the probability of detection and severity of threats. One of our preliminary works [12] 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 [13] presents a novel concept of Attack Fault Tree (AFT), a combination of both attack and fault trees. A model of STA [14] alleviates some assumptions made in timed automata and provides advantages such as choice of transitions requiring satisfaction of very precise clock constraints. Timed automata [15] 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) [16], 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 [17].

3 Preliminaries

3.1 Statistical model checking

Statistical model checking (SMC) is a variation of the well-known classical model checking [18] 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 [19]. In contrast to the classical 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 [15]. In STA, constraints on edges and invariants on locations, such as clocks are used to enable transition from one state to another [13].

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

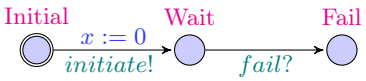


Fig. 2: An exemplar STA

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] $\mathcal{S}_T = \mathcal{S}_{v_1} \,||\, \mathcal{S}_{v_2} \,||\, \ldots\ldots \,||\, \mathcal{S}_{v_n} || Top\_event$ 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 [17]. Detailed syntax and semantics of the SMC queries can be found in [20]. 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

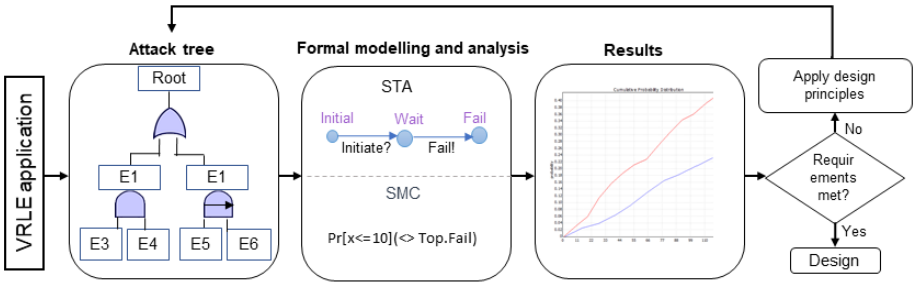


Fig. 3: Proposed framework for security and privacy analysis of social VRLE applications in order to adapt the system design

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 VRLE 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 [21]. We adapt the following three design principles from NIST SP800-160 [7,21] such as: (i) *Hardening* – defined as reinforcement of individual or types of components to ensure that they are harder to compromise or impair, (ii) *Diversity* – defined as the implementation of a feature with diverse types of components to restrict the threat impact from proliferating further into the system, and (iii) *Principle of least privilege* – defined as limiting the privileges of any entity, that is just enough to perform its functions and prevents the effect of threat from propagating beyond the affected component.

4 Proposed framework

In this section, we present details of our proposed framework for security and privacy analysis of social VRLE applications. An overview of the steps followed in our framework is shown in Figure 3. Firstly, we outline the threat scenarios in a social VRLE application [4] using traditional approaches. Secondly, we use attack tree formalism for the modeling procedures. Following this, each attack tree is translated into an equivalent STA to form an NSTA, which is input into the UPPAAL SMC tool. Lastly, we use the quantitative assessment from the tool to determine if the probability of disruption is higher than a set threshold (user specific requirements). Based on this determination, we subsequently adopt the design principles such as: *hardening*, *diversity* and *principle of least privilege* to the generated attack trees. Overall, our framework steps help us to investigate potential design alternatives based on design principles to recommend the best candidate for securing a edge computing based VRLE application.

4.1 Formalization of security and privacy 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 [22]. 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 [13].

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.

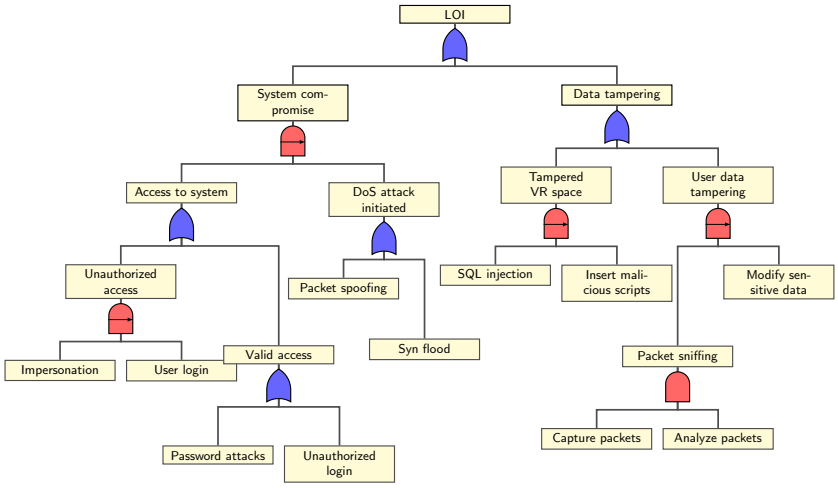


Fig. 4: Attack tree for security factors of VRLE case study viz., vSocial

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 [22] 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

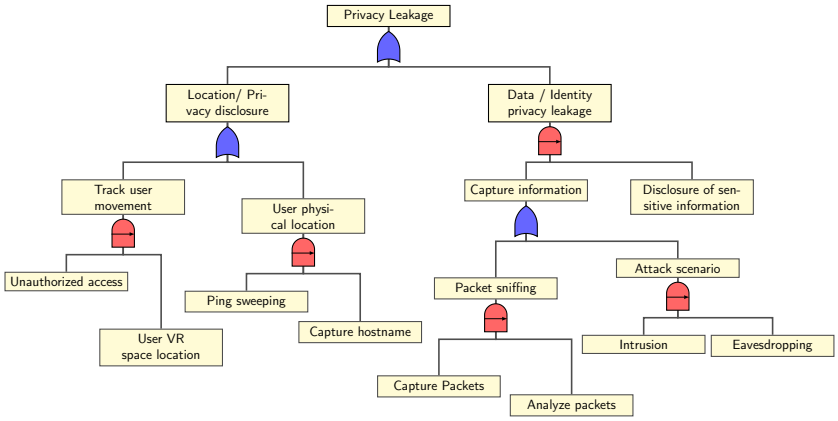


Fig. 5: Attack tree for privacy factors of VRLE case study viz., vSocial

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);

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		

Table 1: Descriptions of leaf nodes in security and privacy attack trees in the VRLE application case study

(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.

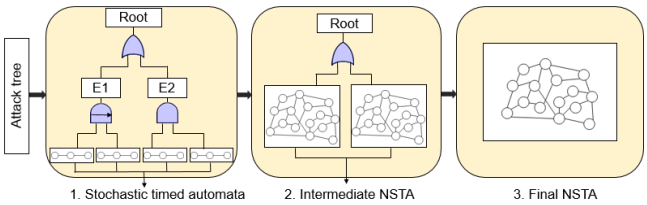
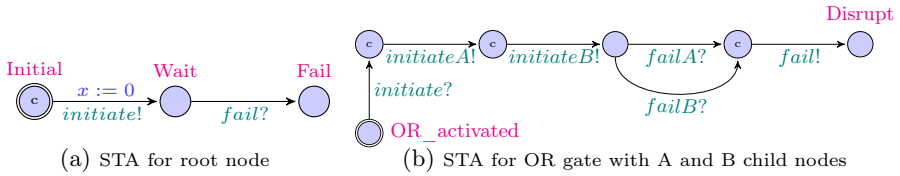


Fig. 6: Logical transformation of compositional aggregation of attack tree models

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.

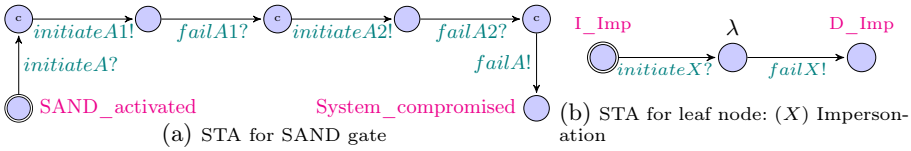
Illustrative example: In the subsequent paragraph, we have explained the above concepts by converting the security attack tree into an exemplar NSTA. Figure 7a shows the converted STA of the LoI i.e., root node (Top_event) equipped with $initiate!$ and $fail?$ signals for the security attack tree. The 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 transition 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.

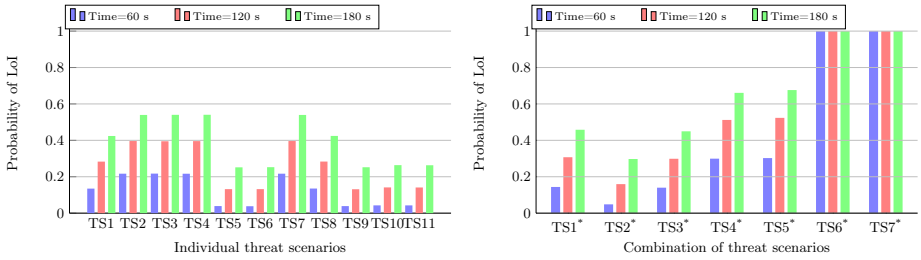
Table 2: Values of λ for the leaf nodes of security and privacy ATs

Security AT		**Privacy AT**	
Security threats	λ	**Privacy threats**	λ
Impersonation	0.006892	Unauthorized access	0.006478
User login	0.0089	User VR space location	0.0094
Password attacks	0.008687	Capture hostname	0.004162
Unauthorized login	0.008687	Ping sweeping	0.002162
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	–	–

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 of parent node is calculated, which propagates upwards in the tree to finally calculate the probability of the root event (LoI). As mentioned earlier, the developed STAs are composed using the parallel composition [15] technique to form an NSTA which is then used for SMC by the UPPAAL tool [5]. Based on the results obtained from SMC, if the design requirements are not met, then we adopt design principles as mentioned in Sections 5 and 6.

5 Quantitative results

In this section, we present the results obtained using our proposed framework. As mentioned in Section 4, the threat scenarios we consider include LoI and privacy leakage for security and privacy attack trees (AT), respectively. In the following analysis, we assume that our design requirement is to keep the probability of LoI and privacy leakage below the threshold of 0.25. For evaluation purposes, we use arbitrary values of λ as parametric input to the leaf nodes as shown in Table 2 obtained from [23], [24]. Note, after providing λ values as parameters to the leaf nodes, we utilize the SMC queries as explained in Section 3 to find the respective probabilities of LoI and privacy leakage. Any other user specified threshold values can also be used in our framework. This is due to the fact that the model checking approach takes the user specified values at the beginning of experiment. For this experiment, we consider LoI (security attack tree) and privacy leakage (privacy attack tree) as goal nodes. In the following set of experiments, we present the obtained probability of the goal nodes with respect to the time window that is used by the attacker.



(a) TS of security AT where - TS2, TS3, TS4, TS7 are the most vulnerable nodes.

(b) TS of security AT where – TS6*, TS7* are the most vulnerable combination

Fig. 9: Vulnerability analysis using the security AT

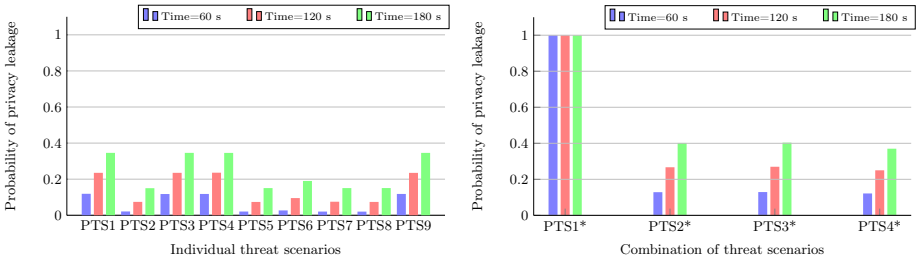
5.1 Vulnerability analysis in the security AT

We assign the values of λ shown in Table 2. However, when assigning a λ value to a leaf node in the attack tree, we consider a very small positive constant $(K) \approx 0.002$ for the remaining leaf nodes. This is because, in real time systems, multiple attack scenarios can happen. To identify a vulnerability in a security attack tree, we analyze: (i) individual leaf nodes, and (ii) combinations of leaf nodes, to determine their effect on the probability of LoI.

i) Individual leaf node analysis: In Figure 9a, we show the probability of LoI over multiple time windows for each leaf node in the security attack tree. We perform a thorough analysis of leaf nodes in the security attack tree for threat scenarios across different time intervals i.e., $t = \{0, 60, 120, 180\}$. For the individual leaf node analysis, the considered threat scenarios (TS) shown in Figure 9a are termed 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 9a, the leaf nodes $TS3$ and $TS4$ (for authorized access) as well as $TS2$ and $TS7$ (for DoS attack) are the most vulnerable in the security attack tree with the maximum probability of 0.53.

ii) Analysis using combination of leaf nodes: Herein, we consider combinations of leaf nodes to identify their impact on LoI. For these experiments, we explore two scenarios: In the first scenario, we consider combinations of leaf nodes that belong to the same sub-tree, and in the second scenario, we consider leaf nodes from different sub-trees.

The considered combination of threat scenarios are enlisted 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, unauthorized login}, $TS7^*$ – {unauthorized login, Sync flood}. As shown in Figure 9b, $TS6^*$ and $TS7^*$ are the most vulnerable combination of threat scenarios with a probability of 1 for an LoI event. As part of further analysis in Section 6, we show how our adoption of design principles on these leaf nodes can enhance the VRLE application resilience against security threats.



(a) TS of privacy AT where - PTS1, PTS3, PTS4, PTS9 are the most vulnerable nodes.

(b) TS of privacy AT where – PTS1* is the most vulnerable combination

Fig. 10: Vulnerability analysis using the privacy AT

5.2 Vulnerability analysis in the privacy AT

We analyze the privacy attack tree for similarly for: (i) individual leaf nodes, and (ii) combinations of leaf nodes. For the individual leaf node analysis in the privacy attack tree, the threat scenarios are termed as: $PTS1$ – unauthorized access, $PTS2$ – capture packets, $PTS3$ – user VR space location, $PTS4$ – ping sweeping, $PTS5$ – analyze packets, $PTS6$ – disclosure of sensitive information, $PTS7$ – intrusion, $PTS8$ – eavesdropping, $PTS9$ – capture hostname. As shown in Figure 10a, the most vulnerable leaf nodes are: $PTS1$, $PTS3$, $PTS4$, $PTS9$ with the highest probability of privacy leakage of 0.34.

For the analysis of combination of leaf nodes, we refer to the combination of threat scenarios as: $PTS1$* – {unauthorized access, user VR space location}, $PTS2$*– {capture packets, disclosure of sensitive information}, $PTS3$* – {unauthorized access, disclosure of sensitive information}, $PTS4$* – {capture packets, analyze packets}. As shown in Figure 10b, $PTS1$* is the most vulnerable combination of threats for privacy leakage with a probability of 1.

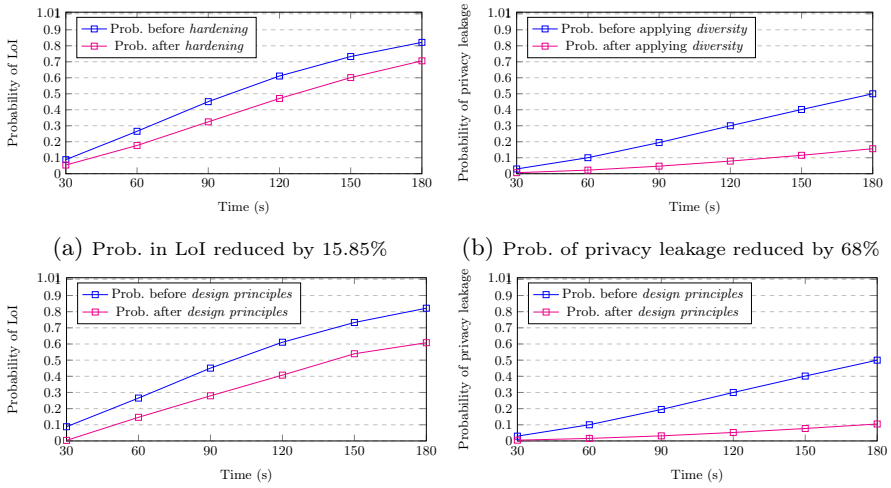
In summary, we can conclude that the above numerical analysis on attack trees can help in identifying LoI and privacy leakage concerns that need to be addressed in the social VRLE design.

6 Recommended design principles

In this section, we present our mitigation approach to improve the trustworthiness of a social VRLE by applying the design principles discussed in Section 3.3 on the identified threat scenarios detailed earlier in Sections 5.1 and 5.2.

i) Implementation of design principles on the security AT: In Figure 11a, we analyze the effect on LoI after applying the *hardening* design principle on password attacks leaf node. When we apply the *hardening* principle, we add new leaf nodes such as firewall and security protocol in the security attack tree. Our results show that the probability of LoI is reduced from 0.82 to 0.69 (15.85% reduction). Similarly, when we apply the *principle of least privilege* to the security attack tree, we observe that the probability of LoI is reduced from 0.82 to 0.79 (3.66% reduction).

ii) Implementation of design principles on the privacy AT: Herein, we again apply the *diversity* design principle on one of the leaf nodes (unauthorized



(a) Prob. in LoI reduced by 15.85%

(b) Prob. of privacy leakage reduced by 68%

(c) Prob. of LoI is reduced by 26.82%.

(d) Prob. of privacy leakage reduced by 80%

Fig. 11: Effect on security and privacy AT after applying design principles

access) in the privacy attack tree. When we apply the *diversity* principle, multiple authentication procedures are added as new leaf nodes. We observe that this significantly reduces the probability of privacy leakage from 0.5 to 0.16 (68% reduction) as shown in Figure 11b. Similarly, when we apply the *principle of least privilege* on the privacy attack tree, we observe that the probability of privacy leakage reduces slightly from 0.5 to 0.48 (4% reduction).

Thus, from the above implementation of individual design principles, we conclude that *hardening* and *diversity* are more effective in reducing the probability of LoI and privacy leakage, respectively. Although, the *principle of least privilege* was incorporated in both trees, there was no significant effect observed on their goal nodes. This observation motivates our next experiment detailed below, where we show the benefits of implementing a combination of design principles to improve the attack mitigation efforts.

iii) Combination of design principles on the security and privacy ATs: To study the effect on probability of LoI and privacy leakage, we adopt a combination of design principles: (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 LoI from 0.81 to 0.6 (25% reduction), and for privacy leakage 0.5 to 0.1 (80% reduction) as shown in Figures 11c and 11d respectively. Thus, we conclude that incorporating a relevant combination of standardized design principles has the potential to mitigate the impact of sophisticated cyber attacks on social VRLE applications. In addition, our above experimental results justify the significance of analyzing the security and privacy threat scenarios for the users of social VRLE applications.

7 Conclusion

Social Virtual Reality Learning Environments (VRLEs) are a new form of immersive VR applications, where security and privacy issues are underexplored. In this paper, we presented a novel framework that quantitatively assesses the security and privacy threat scenarios for a social VRLE application case study viz., vSocial. Specifically, we explored different threat scenarios that possibly cause LoI and privacy leakage in a given social VRLE application. We utilized the attack tree formalism to model the security and privacy threats. We developed relevant attack trees and converted them into stochastic timed automata and then performed model checking using the UPPAAL SMC tool. Furthermore, we illustrated the effectiveness of our framework by analyzing different design principle candidates. We showed a 'before' and 'after' performance comparison to investigate the effect of applying these design principles on the probability of LoI and privacy leakage occurrence. The highlights from our experiments with realistic social VRLE application scenarios indicate that: among the design principle candidates, (i) *{hardening, principle of least privilege}* is the best design principle combination for enhancing security, and (ii) *{diversity, principle of least privilege}* is the best design principle combination for enhancing privacy.

As a part of our future work, we plan to explore the effect of fault and attacks as a combination on social VRLE applications using the attack-fault tree formalism [13]. This will allow us to reason about the safety metrics and study the safety, security and privacy trade-offs in social VRLE applications. Since different components in a typical social VRLE application go through different maintenance actions, we also plan to explore the impact of various maintenance strategies on the reliability and availability metrics of social VRLE applications using the fault maintenance tree formalism [25].

Acknowledgements

This work was supported by the National Science Foundation under awards: CNS-1647213 and CNS-1659134. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

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. S. Bruce, "Attack trees", *Dr.Dobb's journal*, 24.12, 21-29, 1999.
7. "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`
8. R. Roman, J. Lopez, M. Mambo, "Mobile Edge Computing, Fog etal.: A Survey and Analysis of Security Threats and Challenges", *Elsevier FGCS*, 2016.
9. S.Yi, Z.Qin, Q.Li, "Security and Privacy Issues of Fog Computing: A Survey", *Proc. Intl. Conf. Wireless Algorithms, Systems, Applications*, 2015.
10. M. A. Khan, K. Salah, "IoT security: Review, Blockchain Solutions, and Open Challenges", *Elsevier Future Generation Computer Systems*, 2018.
11. E. Byres, M. Franz, D. Miller, "The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems", *IEEE Conf. International Infrastructure Survivability Workshop*, 2004.
12. 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.
13. R. Kumar, M. Stoelinga, "Quantitative Security and Safety Analysis with Attack-Fault Trees", *IEEE 18th Int. Symposium on HASE*, 2017.
14. N. Bertrand, P. Bouyer, T. Brihaye, Q. Menet, C. Baier, M. Grosser, M. Jurdzinski, "Stochastic Timed Automata", *Logical Methods in Comp. Sci.*, 2014.
15. P. Ballarini, N. Bertrand, A. Horvath, "Transient Analysis of Networks of Stochastic Timed Automata Using Stochastic state classes", *Springer*, 2013.
16. A. Aziz, K. Sanwal, V. Singhal, R. Brayton, " Model-checking Continuous-time Markov chains", *ACM Transactions on Computational Logic*, 2000.
17. D. Alexandre, K. Larsen, A. Legay, M.Mikučionis ,D. Poulsen, "Uppaal SMC Tutorial", *Int.J. on Software Tools for Technology Transfer*, 2015.
18. E. M. Calrk jr, O. Grumberg, D. Peleg, "Model checking", *MIT Press*, 2000.
19. 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.
20. 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.
21. A. Laszka, W. Abbas, Y. Vorobeychik, X. Koutsoukos,"Synergistic Security for the Industrial Internet of Things: Integrating Redundancy, Diversity, Hardening", *IEEE ICII*, 2018.
22. G. Norman, D. Parker, J. Sproston, "Model checking for probabilistic timed automata", *Formal Methods in System Design*, Vol. 17, pp. 164–190, 2013.
23. P. Saripalli, B. Walters, "QUIRC: A Quantitative Impact and Risk Assessment Framework for Cloud Security", *IEEE 3rd Int. on Cloud Computing*, 2010.
24. 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.
25. 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.