GRAVITAS: Graphical Reticulated Attack Vectors for Internet-of-Things Aggregate Security

Jacob Brown, Tanujay Saha, and Niraj K. Jha *Fellow, IEEE*

Abstract—Internet-of-Things (IoT) and cyber-physical systems (CPSs) may consist of thousands of devices connected in a complex network topology. The diversity and complexity of these components present an enormous attack surface, allowing an adversary to exploit security vulnerabilities of different devices to execute a potent attack. Though significant efforts have been made to improve the security of individual devices in these systems, little attention has been paid to security at the aggregate level. In this article, we describe a comprehensive security model, called GRAVITAS, for IoT/CPS that can identify their security vulnerabilities and optimize the placement of defenses within the system. While existing defense mechanisms consider only known vulnerabilities, our model employs a machine learning approach to extrapolate undiscovered vulnerabilities, enabling us to identify attacks overlooked by manual penetration testing. The model is flexible enough to analyze practically any IoT/CPS and provide the user with a concrete list of defenses that can reduce system vulnerability at minimum cost. GRAVITAS can be employed by governments, companies, and individual users to design secure IoT/CPS without the time and expense of traditional pen-testing, providing a measure of efficiency and security in a world where IoT/CPS devices will soon be ubiquitous.

Index Terms—Attack Graphs; Cyber-physical Systems; Cybersecurity; Internet-of-Things; Machine Learning; Network Security.

1 Introduction

Internet-of-Things (IoT) refers to any system that comprises multiple connected devices that provide transmission or computational services as one networked entity [1]. Cyber-physical systems (CPSs) employ sensor data to monitor the physical environment and create real-world change using actuators. These broad categories include systems ranging from a single Bluetooth-enabled smartwatch to a "smart city" containing millions of devices. Many of these devices employ rudimentary operating systems and are energy-constrained, making even basic security features too costly to implement. IoT/CPS may also consist of a diverse set of devices and complex network topologies, presenting a large attack surface that provides multiple enticing opportunities for a cunning adversary.

By 2021, global spending on IoT is projected to reach $520 billion [2]. This underscores how ubiquitous these systems will soon become. Over the next few years, we can expect to see IoT systems become commonplace in healthcare, manufacturing, transportation, law enforcement, energy distribution, and many other applications not yet imagined [1]–[8]. IoT systems are projected to significantly reduce costs and improve efficiency across many industries, encouraging mass adoption by both corporations and governments [2], [9]. The development of 5G communication infrastructure, autonomous vehicles, and hardware specifically designed for machine learning (ML) is also accelerating this process [9].

However, many industry experts and leading political figures argue that the widespread adoption of IoT systems has the potential to engender "catastrophic" consequences [10]–[12]. One ominous sign is the Mirai botnet attack of 2016, a distributed denial-of-service (DDoS) attack that briefly brought down large parts of the Internet on the U.S. East Coast [13]. This attack was particularly notable because a single security flaw, i.e., unchanged default passwords, resulted in significant technological and economic disruption. This catastrophic outcome highlights how a single malicious adversary can potentially compromise an entire IoT/CPS, if not the whole Internet [13].

Cyber attacks like the Mirai botnet should serve as a warning: every IoT/CPS must be scrutinized for vulnerabilities before deployment. The large attack surface of autonomous vehicle networks, smart cities, and other publicly-accessible IoT/CPS should draw particular scrutiny because a security breach of at least one connected device is nearly inevitable, allowing the adversary to wreak havoc on other parts of the system. In order to prevent future Mirai-like attacks, engineers will need to take into account not just the security of individual devices, but the security of the system as a whole.

Moreover, securing IoT/CPS is challenging because of the limited resources available to their constituent devices. Such limitations often preclude the devices from employing intrusion detection mechanisms and executing complex cryptographic protocols. Although IoT-friendly lightweight protocols exist [14], [15], it is still challenging to select a suitable combination of defenses to obtain optimal performance and security of the system.

Our model, called GRAVITAS, overcomes these challenges by combining the hardware, software, and network stack vulnerabilities of a system into a single attack graph. This attack graph, which also includes connections between

This work was supported by NSF under Grant No. CNS-1617628. *Jacob Brown, Tanujay Saha, and Niraj K. Jha are with the Department of Electrical Engineering, Princeton University, Princeton, NJ, 08544 USA, e-mail:{jacob.brown,tsaha,jha}@princeton.edu.*

vulnerabilities as predicted by an ML model, allows us to derive a probabilistic "vulnerability score" that describes an attack's appeal to different classes of adversaries. We then add a subset of user-defined defenses to the device using an optimization process that lowers the vulnerability score at minimum cost. With an IoT/CPS as its input and a list of the most feasible defenses as its output, GRAVITAS presents a security model that allows the user to discover new vulnerabilities and proactively defend the system before its deployment at minimum cost.

The article is organized as follows. Section 2 describes related work that informs this article. Section 3 presents background material on the ML-generated attack graph that serves as an inspiration for GRAVITAS, as well as a primer on attack graphs and the CVSS scoring methodology. Section 4 provides a brief description of how GRAVITAS provides novel capabilities to IoT/CPS designers. Section 5 gives details of our methodology, including the user inputs and model outputs. Section 6 provides a practical example of the model's functionality. Section 7 includes a discussion and ideas for future work. Section 8 concludes the article.

2 Related Work

Most IoT-related security research to date has focused on remediating device-specific or application-specific security vulnerabilities. Over the last decade, researchers have discovered eavesdropping on implanted medical devices, "outage" attacks on IoT systems in nuclear power plants, tampering with smart home devices, identity theft using corrupted RFID tags, and poisoning ML models by changing sensor data, among many others [16]–[20]. This research also occurs in the corporate world: IBM, like several other companies that offer a cloud-based IoT platform, operates a lab specifically dedicated to pen-testing IoT systems; the company claims that its laboratory has discovered over 1000 new vulnerabilities since 2017 [21]. As IoT/CPS devices increase both in number and in variety, we can expect the number of articles that discover security flaws to continue increasing.

Researchers have also developed models specifically for IoT systems. Ur-Rehman et al. describe an attack graph model for IoT systems that employs probabilistic notions of vulnerability to gauge the "weakest link" in a complex system [22]. However, the authors model each device as a single node and do not take into account the complex topology of vulnerabilities that exists within each device. Other tools, such as MulVal, TVA, TAG, and A2G2V, model both the internal vulnerabilities of connected devices and their network configuration in an attack graph format, but do not consider the unique (and often undiscovered) vulnerabilities of public-facing IoT/CPS devices or the convoluted attack paths available to a clever adversary [23]–[26]. Commercial network security software, such as Naggen, is usually based on a similar organizing principle [27]. GRAVITAS seeks to merge the best aspects of all these approaches, creating an IoT/CPS model that recognizes the hidden vulnerabilities of each device while ensuring that the network's topology and access control parameters are accurately taken into account.

3 Background

GRAVITAS employs several concepts developed in previous network security models. These include the SHARKS IoT/CPS model, attack graphs, and the Common Vulnerability Scoring System (CVSS). This section provides an introduction to these concepts.

3.1 The SHARKS Framework

GRAVITAS builds on the work of SHARKS (Smart Hacking Approaches for RisK Scanning in Internet-of-Things and Cyber-Physical Systems based on Machine Learning), which provides a novel framework for predicting IoT/CPS vulnerabilities [28]. Instead of artificially separating a system into different layers, SHARKS eschews a rigid classification and models an attack as it appears to an adversary: a series of steps that begins at an "entry point" (a root node) and ends at a "goal" (a leaf node). The SHARKS attack graph (Fig. 1) was created by deconstructing 41 known attacks on IoT/CPS into a series of steps represented by a node chain (attack path), and subsequently merging every node chain into a single directed acyclic graph (DAG). This graph makes no distinctions between network-level, hardware-level, or software-level nodes: what matters is the procedure that brings attackers to their desired destination.

The DAG has 51 paths from a root node to a tail node, but only 41 attacks were used to construct it. This discrepancy means that SHARKS found 10 novel exploits, all of which were subsequently validated in a real-world system. SHARKS then assigns descriptive features to each node in the attack graph, using one-hot encoding for categorical features as well as continuous features such as the node's mean height in the graph. The authors assigned a target feature to each pair of nodes indicating whether the two were connected by an edge (an ordered sequence of two steps in an attack). This dataset was used to train a Support Vector Machine (SVM) model, which "discovered" 122 new security vulnerabilities.

3.2 Attack Graphs

Both SHARKS and GRAVITAS are based on attack graphs, albeit with minor differences in their node types. We describe each graph (G) using the following terminology:

- N: The set of nodes in the graph. Each node represents a single vulnerability in the system.
- E: The set of edges in the graph. Unlike in other attack graph models, edges do not have an access control parameter; each edge instead represents a possible path between vulnerabilities. Different permissions are instead represented by different nodes (see Section 5.2.2).
- D: The set of nodes and edges corresponding to one device. Every device is depicted by a subgraph of the complete attack graph ($D \subset G$).
- A: The nodes at which an adversary can access the system. These "entry nodes" are the starting point for any attack. They are also vulnerabilities ($A \subset N$).
- L: The nodes at which an adversary completes an attack. These "attack outcomes" represent the end goal of an adversary's attack. They are not vulnerabilities

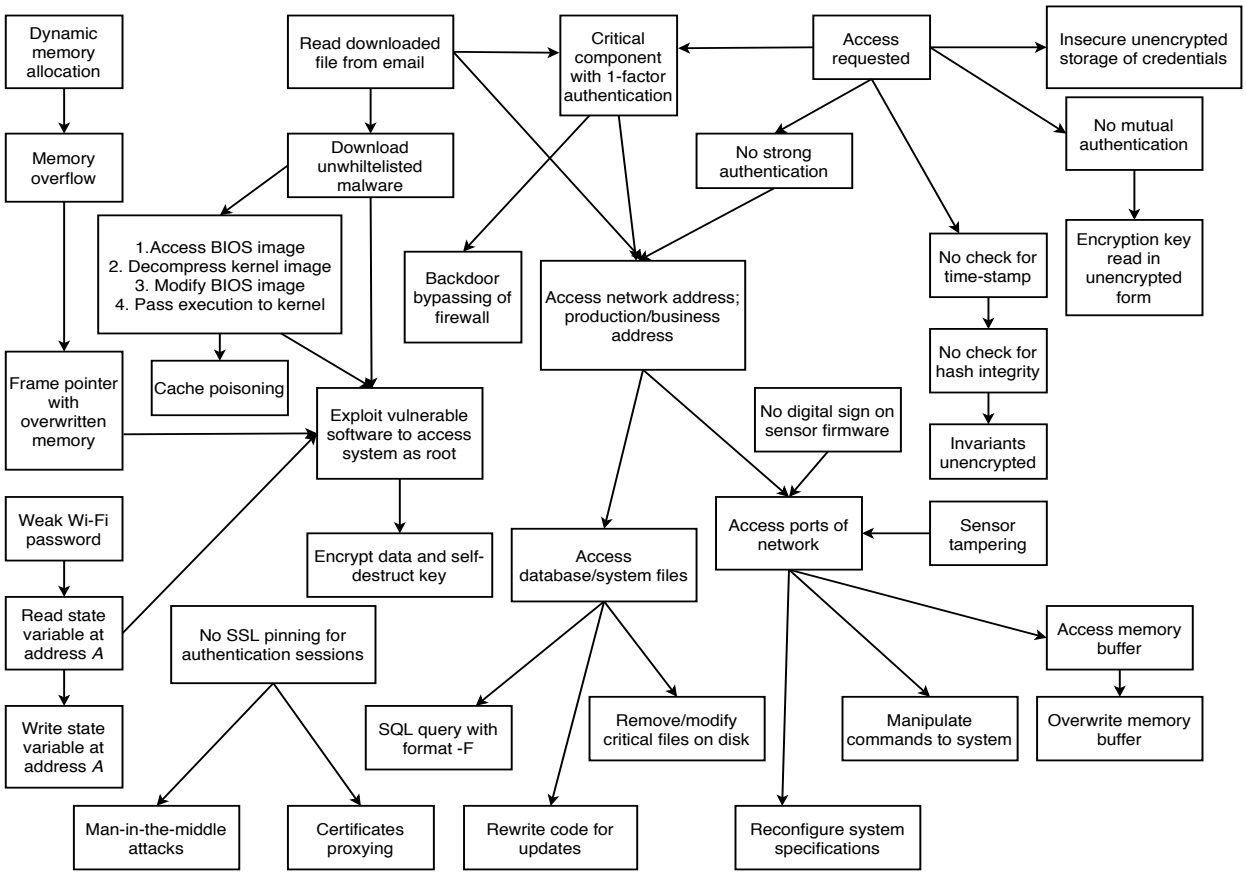


Fig. 1: The original SHARKS attack graph [28]

per se, but are denoted as vulnerabilities for the sake of simplicity ($L \subset N$).

- P: The set of nodes that constitute a complete attack. Each "attack path" begins at an entry node and concludes at an attack outcome node. More formally, an attack path P is any ordered set of nodes in the form $\{a, n_1, ..., n_k, l\}$ where $a \in A$, $n_i \in N$, $l \in L$. The same entry node and attack outcome pair can be a part of multiple attack paths.
- M: The defenses that can be applied to the graph.
- $\widetilde{M}$: The subset of defenses chosen by the optimization process.

3.3 Common Vulnerability Scoring System (CVSS)

Every node in the attack graph is assigned an intrinsic vulnerability score (see Section 5.3.1). This score is calculated using a formula similar to FIRST's Common Vulnerability Scoring System (CVSS) [29]. Some of the scores in the *Attack Vector* and *Attack Complexity* categories are taken from an article by Ur-Rehman et al. that adapts some of the generic CVSS scores to better reflect IoT devices [22]. Each node's intrinsic vulnerability score is calculated using the factors described in Table 1. There are three principal categories of factors: *exploitability*, *impact*, and *defense*. *Exploitability* refers to the effort required by an adversary to "succeed" in an attack step, while *impact* refers to the damage that a successful attack can inflict on the security of the system. *Defense* refers to the extent to which the attack is prevented from being exploited. The scores in the *exploitability* category are determined by the composition of the attack graph and are computed algorithmically, while those for *impact* and *defense* are decided by the user based on their judgement of an attack's impact and their choice of defenses.

4 Motivation

While SHARKS is able to find new attacks on specific kinds of IoT/CPS, it is too generic to adequately model the multitude of intricate attack pathways in a real-world system. GRAVITAS solves this issue by creating a unique attack DAG for every device in an IoT/CPS and adding additional attack paths between devices based on network topology. The goal is not to find specific weaknesses in each device (a drawn-out process that often requires extensive manual pen-testing) but to understand the effects that a compromised device would have on the vulnerability of the entire system. Moreover, unlike other IoT/CPS security models, GRAVITAS can account for vulnerabilities that have not yet been discovered, which enables a proactive approach to network security. GRAVITAS can also suggest defenses that reduce system vulnerability at the lowest cost, employing a "defense-in-depth" optimization approach that is intractable to human analysis. By giving an administrator the ability to visualize and repair IoT/CPS vulnerabilities before it is deployed, GRAVITAS hopes to prevent the next Mirai-like attack before it happens.

TABLE 1: CVSS scoring values [29]

Category	Factor	Type	Score
Exploitability	Attack vector	Network	0.85
		Adjacent	0.62
		Local	0.60
		Physical	0.44
	Attack complexity	Low	0.77
		Medium	0.44
		High	0.20
	Scope	Changed	N.A.
		Unchanged	N.A.
	Privileges required	None	0.85
		Low; Scope Changed	0.68
		Low; Scope Unchanged	0.62
		High; Scope Changed	0.50
		Low; Scope Unchanged	0.27
	User interaction	None	0.85
		Required	0.62
	Accessibility	High	0.80
		Medium	0.60
		Low	0.40
		None	0
Impact	Confidentiality Integrity Availability	High	0.56
		Low	0.20
		None	0
Defense	Node & edge defense	None	1
		Workaround	0.90
		Temporary	0.60
		Definite	0.30
		Infallible	0

5 Methodology

Next, we describe the GRAVITAS methodology. As shown in Fig. 2, GRAVITAS consists of four primary components:

1) Deriving the device templates from SHARKS.
2) Creating an attack graph from the devices and network topology specified by the user.
3) Calculating the intrinsic and cumulative vulnerability scores for every node in the graph.
4) Optimizing the placement of defenses to reduce the total vulnerability of the system.

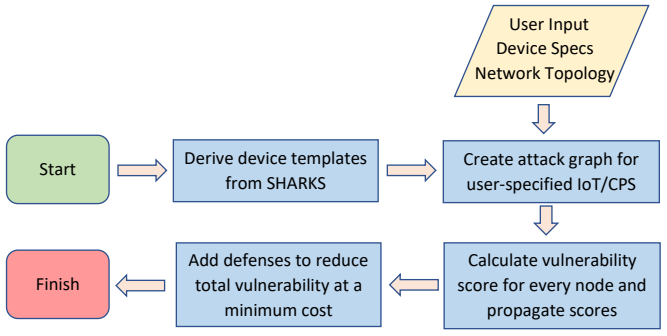


Fig. 2: The overall structure of GRAVITAS

These components are described in Sections 5.1 through 5.4, respectively. Section 5.5 describes the adversary model, whereas Section 5.6 describes a quality assurance program that employs randomly-generated "Smart City" IoT/CPS to test the robustness of the internal parameters of GRAVITAS.

5.1 Deriving Device Templates

All device attack graphs used in GRAVITAS are derived from an updated version of the SHARKS graph. This master attack graph template (J) consists of the original SHARKS graph, including ML-predicted edges between nodes that indicate new vulnerabilities, in addition to a new set of nodes designated as attack outcomes, L. Table 2 lists the attack outcomes; they collectively represent all of the IoT/CPS attacks described in [1]. We also designate certain nodes from the original SHARKS graph as entry nodes, A. The master attack graph J is still a DAG, ensuring that any attack paths P derived from it will be finite in length and non-repeating. Figs. 3 and 4 show a template for a sensor alongside a system-specific sensor from the "Smart Home" system described in detail in Section 6.

TABLE 2: The attack outcomes in the master attack graph template J

Attack Outcome	Example
Eavesdropping over network	Unencrypted communication channel allows adversary to glean information
Denial-of-Service (DoS)	Mirai botnet (Infected devices spam DNS servers)
Disabling device	Adversary remotely turns off a smart city drone mid-operation, causing it to crash
Actuator malfunction	IoT-enabled pacemaker told to increase pace of electric shocks, causing serious harm to the patient
Data leak	Adversary accesses memory of a smart lock, learning the times of day when the victim is not at home
Data change	Adversary gains control of a traffic sensor and provides fake data to an ML algorithm that suggests routes for autonomous vehicles
Replay attack	Communications protocol does not use a digitally signed timestamp, allowing the adversary to resend previous commands
Ransomware attack	Adversary gains root access to device and encrypts essential files, asking the user to present payment in exchange for the key
Obtain authentication key to device i with permissions j	Adversary finds password on a mobile device that permits him to login to a local controller from a different device
Obtain open access to device i with permissions j	Adversary finds password on a mobile device that permits him to login to a local controller directly from the mobile device

5.1.1 *Creating the Device Templates*

Every device template T_i is a subgraph of J, the master attack graph ($T_i \subset J$). A device graph D consists of the corresponding device template with modifications specified by the user input. While each device graph D derived from a template is a DAG, the complete system attack graph G may contain cycles in certain system topologies. This is acceptable because the "long" cycles created by pathways between devices still permit the cumulative vulnerability score calculation process to converge (see Section 5.3.2 for further details).

To create a device graph D, we must first classify the device using four factors. The first of these factors, which we refer to as a *category*, describes the purpose of the device; Fig. 5 shows a flowchart that helps the user decide which of the six categories to select. We refer to the other three factors as *subcategories*, each of which describes a physical limitation

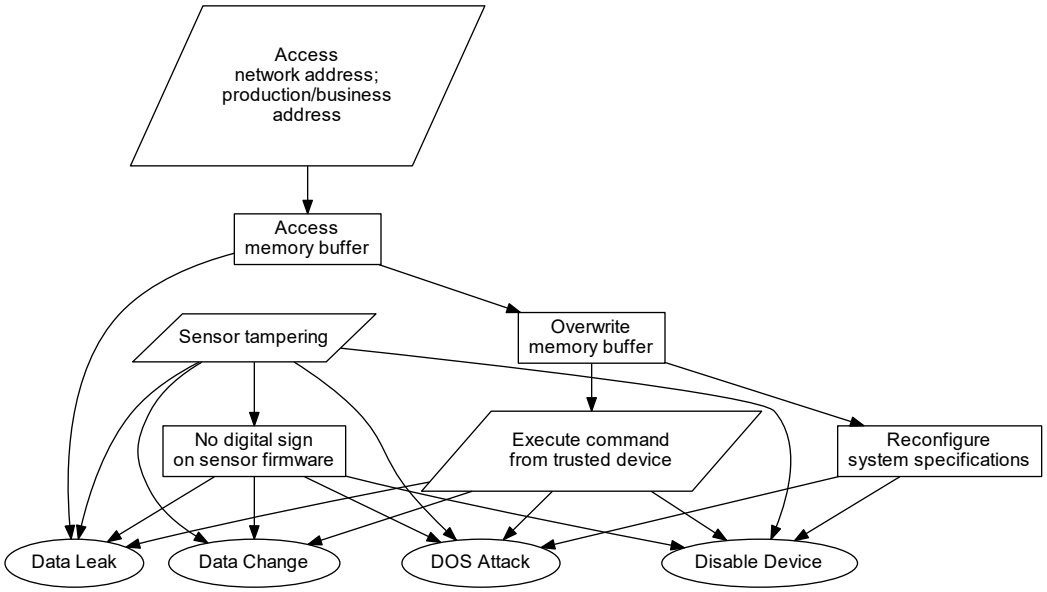


Fig. 3: The template graph for a Sensor (Non-updatable, Send and Receive, Local Network Access). Ovals represent attack outcomes and parallelograms represent entry nodes.

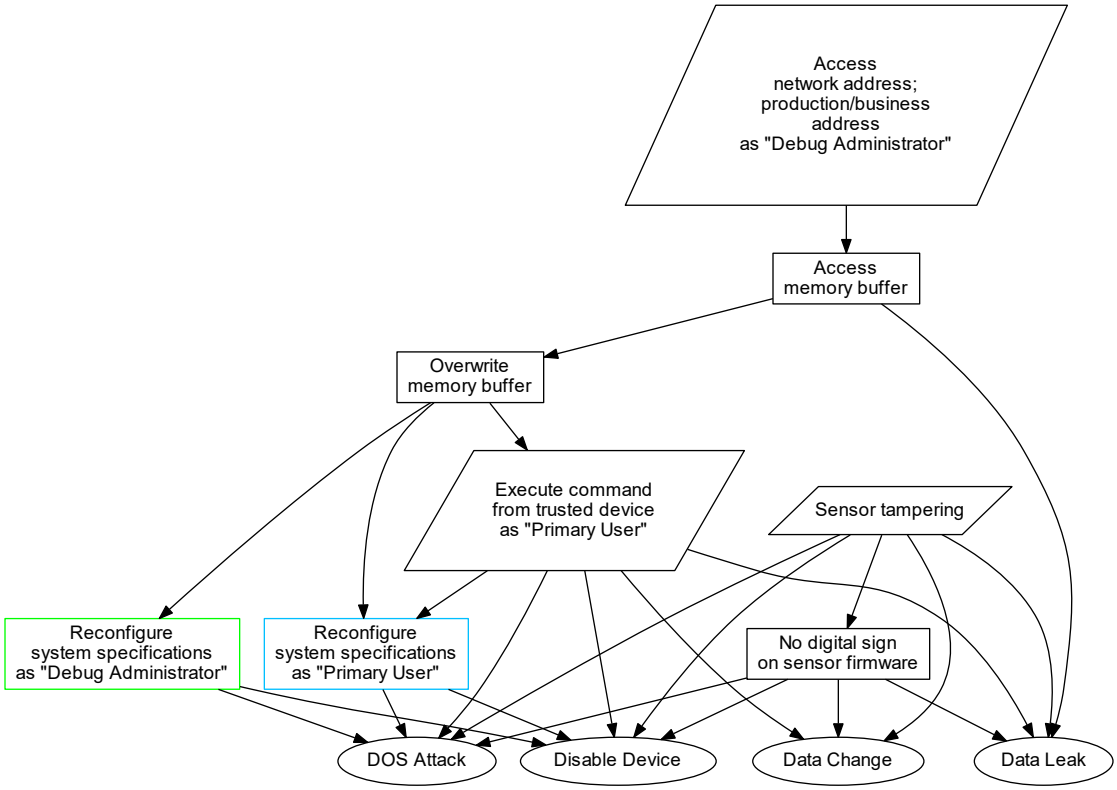


Fig. 4: Nest Garden Sensor in the ”Smart Home” system as derived from the template in Fig. 3. The green and the blue nodes represent the same subgraph that is repeated for each user-specified permission (see Section 5.2.2 for more detail).

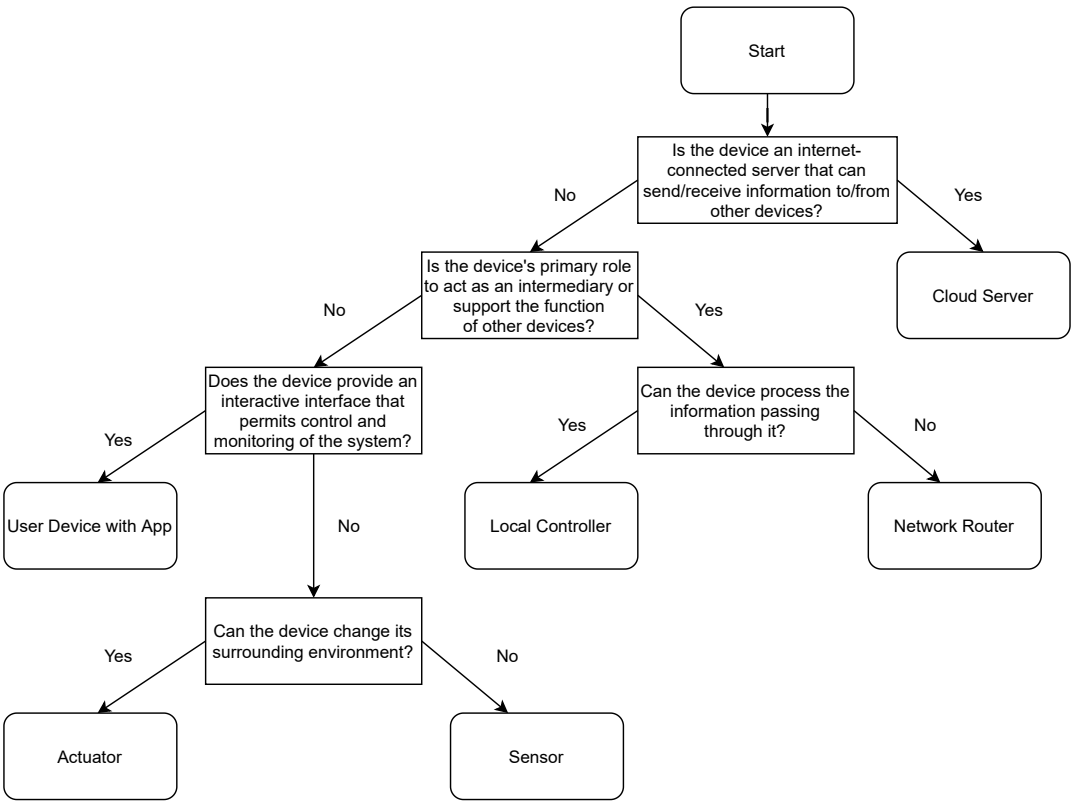


Fig. 5: Flowchart for choosing the correct device category

TABLE 3: Device subcategories for template construction

Factor	Subcategory	Description	Examples
1	Non-updatable	Device's application-level software and firmware cannot be changed	Devices with application software as immutable firmware, devices without a CPU (including some ASICs), devices without transistors or memory
	Updatable	Device's application-level software or firmware can be changed	Any device that runs a version of Linux or most versions of RIoT, PSoC and FPGA-like devices
2	Local net- work access	Device is only connected to a local network (no direct Internet access)	Fitbit Versa 2 (still considered local network access only if Internet-based data is accessed via a non-router proxy device)
	External Net- work access	Device is connected to the Internet (this can be through an adjacent router)	Apple Watch Series 5 Cellular Model
3	Send	Device is physically capable of broadcasting a signal in a format readable to other devices	A sensor with an output port containing a pin that can vary its voltage
	Receive	Device is physically capable of receiving and "under-standing" a signal from another device	An actuator with a simple electronic circuit that depends on the input voltage level
	Send and re- ceive	Device can both send and receive information	4G-enabled smartphone, drone with camera

of the device and consequently a hard boundary on the kind of attacks it is susceptible to. These subcategories are described in Table 3. The categories and subcategories together provide comprehensive coverage of the numerous IoT/CPS applications listed in [1].

5.2 Creating User-Specified IoT/CPS

Fig. 6 provides an overview of how GRAVITAS creates an attack graph G from user input. In addition to specifying inherent device properties, such as the device category and subcategories, the user must also describe each device's location in the network topology. This includes its connections (wired and wireless) to devices in the local network as well as its ability to connect to an external network such as the Internet. Table 4 describes a subset of these device-level characteristics. Note that most of these characteristics are entirely optional. Only "Name," "Category," and "Subcategory" are strictly necessary; all other characteristics receive default values that treat the device as a disconnected component with a low security risk. For an example of a user-specified system, see Section 6.

5.2.1 Defenses

The user specifies M, the set of potential defenses that can be applied during optimization. Table 5 describes these properties in detail. Defenses can act at two places: nodes and edges. A *node defense* changes the score used in the node's intrinsic vulnerability score calculation (see Section

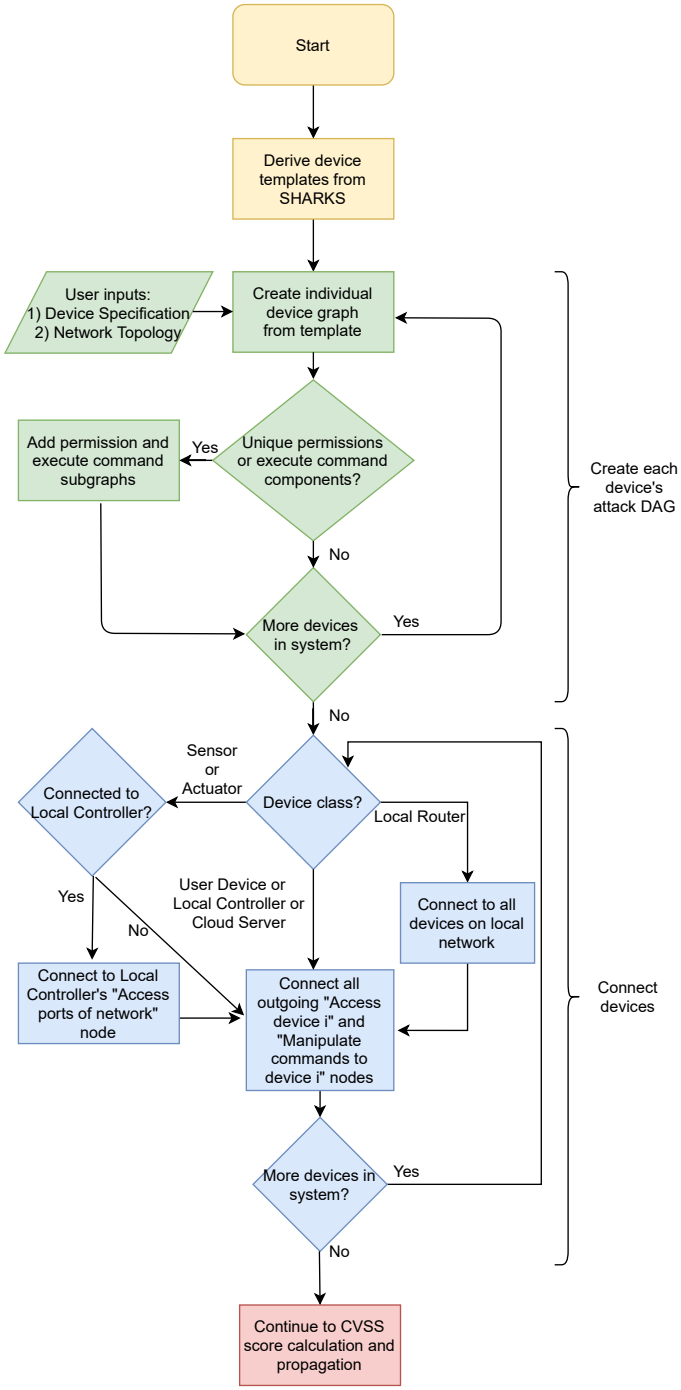


Fig. 6: The procedure for creating the complete system graph G from user input

5.3.1). An *edge defense* changes the weight multiplier of the corresponding edge during cumulative vulnerability score propagation (see Section 5.3.2). The cost of each defense and overall improvement in system security due to it are used in the objective function that determines which defense to add (see Section 5.4.1).

Every device graph D initially gives each of the device's permissions blanket access to a device's capabilities, similar to an Administrator permission. To limit the access of a certain permission, the user can specify a defense that removes an edge between nodes in a permission subgraph (see

TABLE 4: User-specified device properties

Device property	**Description**
Name	Unique name of device
Category	Choose from the categories in Fig. 5
Subcategory	Choose from the subcategories in Table 3
Device set	Devices in the same set are treated as identical; during system optimization, all devices in a single set are given the same defense concurrently
Confidentiality Scores	A list of CVSS confidentiality scores for the attack outcomes of this device
Integrity scores	A list of CVSS integrity scores for the attack outcomes of this device
Availability Scores	A list of CVSS availability scores for the attack outcomes of this device
Accessibility scores	A list of accessibility scores for the entry nodes of this device
Login Permissions	A list of the devices that possess the credentials to login to this device; a single device may have multiple permission types, and multiple devices may login to this device using the same permission type
Execute command permissions	A list of the devices that possesses the credentials to send valid commands to this device; a single device may have multiple permission types, and multiple devices may login to this device using the same permission type
General permissions	Some simple devices, such as embedded IoT sensors, do not possess a formal security protocol, and can be controlled by any device that connects to them
Local networked devices	A list of the devices in this device's local network; if this is a "hub" network centered around a device like a WiFi router, the algorithm will automatically connect all devices in a local network group as long as each device is connected to the hub

TABLE 5: User-specified defense properties

Defense Property	**Description**
Defense Name	Unique name for the defense; all defenses in a device set that share the same name will be applied simultaneously if chosen during optimization
Device Name	A list of the devices to which this defense can be applied
Cost	The cost of the defense in relative units
Node Score List	A list containing each node and updated *node defense score* affected by the defense
Edge Score List	A list containing each edge and updated *edge defense score* affected by the defense

Section 5.2.2). This "negative list" approach simplifies the user input and ensures that low-cost/high-impact defenses are added at the beginning of the optimization process.

5.2.2 Permission Subgraphs

GRAVITAS allows the user to specify permissions for every device. Unlike other attack graph models, the access permissions are each represented by a separate copy of a subgraph rather than as a logical statement at certain nodes or edges [30]–[32]. This design choice makes a visualization of the attack graph easier to understand, and also simplifies the calculation of cumulative vulnerability scores because we can apply the same closed-form calculation to every node-edge pair (see Section 5.3.2).

We model two different types of permissions: login permissions and execute command permissions. With a login

permission, a user with the correct credentials can execute any (permitted) command on the system; this is similar to a user profile on a Linux or Windows system. With an execute command permission, a user with the correct credentials can execute a (permitted) command from a specified list. For example, this could be a set of commands recognizable in a JavaScript Object Notation (JSON) packet or the movement controls for an autonomous drone. Depending on the configuration of the IoT/CPS, different devices may login/execute commands under the same permission name.

As described in Table 6, certain nodes in every device template are associated with login permissions, execute command permissions, or both. These subgraphs are "repeated" for every permission type of that device. Fig. 3 shows the template for a sensor, while Fig. 4 shows the system-specific device graph D with nodes in different permission subgraphs delineated in different colors. Note that the user can specify defenses that effectively remove certain nodes or edges, allowing the user to set restrictions on what each permission can access.

TABLE 6: Permission subgraph node types

Attack node	**Login Permission**	**Execute Command Permission**
Access network address; production/business address	Yes	No
Access ports of network	Yes	No
Reconfigure system specifications	Yes	Yes
Access database/system files	Yes	Yes
SQL query with format -F	Yes	Yes
Rewrite code for updates	Yes	Yes
Remove/modify files on disk	Yes	Yes
Download unwhitelisted malware	Yes	Yes
Read state variable at address A	Yes	Yes
Write state variable at address A	Yes	Yes
Execute command	No	Yes

5.2.3 Connecting the Devices

Once every device's attack graph D has been produced, we can connect them together into an aggregate attack graph G using the network topology specified by the user. A login permission j is represented by an edge originating from the attack outcome "Obtain authentication key to device i as permission j," while an execute command permission j is represented by an edge originating from from "Manipulate commands to device i with permission j." Both lead to the node "Access network address, production/business address as permission j" of device i. Both of these connections allow the devices to bypass the device's authentication procedures, meaning that the attacker does not have to start at the "Access requested" node where almost all non-authenticated adversaries must begin their attack. For certain sensors and actuators, "Sensor tampering" and "No digital sign on sensor firmware" are connected to the "Access ports of network" node of the neighboring local controller; this represents the local network that exists between certain local controller and sensor/actuator setups, such as those involving an Arduino. To model access to a local network, every router's "No strong authentication" node is connected in both directions to that same node in all other adjacent routers, and is also connected to the "Access requested" node for adjacent devices that are not routers. External network access (i.e., to the Internet) is modeled by including nodes such as "Download unwhitelisted malware" in the device template.

5.3 Calculating and Propagating Vulnerability Scores

Every node in the attack graph is first assigned an intrinsic vulnerability score. These scores are then "propagated" through the graph, giving each attack node a cumulative vulnerability score. The total vulnerability of IoT/CPS, which we call an *adversary score*, is calculated using the adversary model chosen by the user and involves a function of the cumulative vulnerability scores of entry nodes (see Section 5.5). This score is used in the objective function employed in the defense placement optimization process described in Section 5.4. The intrinsic vulnerability scores are only calculated once, whereas the cumulative vulnerability scores (and adversary score) must be recalculated after adding a new defense.

All intrinsic and cumulative vulnerability scores fall into the [0,1] range. This is a departure from the traditional CVSS scoring range of [0,10], but it allows us to treat each score as the probability that an adversary will attempt the attack and succeed in exploiting it. This approach is widely used in attack graph models because it allows for a probabilistic understanding of an adversary's movement through the graph [22], [32], [33].

5.3.1 Calculating Intrinsic Vulnerability Scores

Algorithm 1 gives the equations for calculating the intrinsic vulnerability score. These equations (including their constants) are practically identical to the original CVSS equations, but with some minor modifications that ensure nodes n_i, n_j, n_k in the middle of a path P are not treated as attack outcomes or "zeroed out" if the user specifies no impact scores for those nodes [29].

Each node's intrinsic vulnerability score is calculated using the factors described in the **Data** section of Algorithm 1. The value of each factor is computationally determined by an algorithmic version of the corresponding flowchart on pages 20-21 of the CVSS v3.1 user guide, and is thus dependent on the node's permission characteristics and its topographic position within the device graph D [29]. The $confidentiality$, $integrity$, and $availability$ values of attack outcome nodes are set by the user, as is the $accessibility$ score of each entry node. Note that $n \notin (A \cup L)$ by definition must have $confidentiality$, $integrity$, $availability$, and $accessibility$ scores of 0.

5.3.2 Propagating Cumulative Vulnerability Scores

To model IoT/CPS security, we need an understanding of how intrinsic vulnerability scores of different nodes interact. This interaction is represented by each node's cumulative vulnerability score, which is calculated using an equation that involves the cumulative vulnerability scores of adjacent child nodes (see Algorithm 2). Since our system graph G may contain cycles, this score "propagation" may need to be performed on the system multiple times until the cumulative vulnerability scores converge.

Algorithm 1: Intrinsic Vulnerability Score Calculation

Data: A node's $scope$, $attack\_Vector$, $attack\_Complexity$, and $user\_Interaction$ scores, which are determined algorithmically; $confidentiality$, $integrity$, $availability$ for attack outcome nodes; and $accessibility$ for entry nodes
Result: A node's intrinsic vulnerability score
$IVS = 1 - [(1 - confidentiality)(1 - integrity)(1 - availability)(1 - accessibility)]$;
if *scope* is *unchanged* **then**
  $Impact = 6.42 \times ISS$;
else
  $Impact = 7.52 \times (ISS - 0.029) - 3.25 \times (ISS - 0.02)^{15}$;
end
$Exploitability = 8.22 \times attack\_Vector \times attack\_Complexity \times privileges\_Required \times user\_Interaction$;
if $Impact < 0$ **then**
  $Impact = 0$;
end
if *scope* is *unchanged* **then**
  $x = min[(Impact + Exploitability), 10]$;
else
  $x = min[1.08 \times (Impact + Exploitability), 10]$;
end
$intrinsic\_Score = x \times defense\_Score$

Similar to a recurrent neural network (RNN), we need to update the cumulative vulnerability score at each node using the cumulative vulnerability scores of previous iterations. Given the cyclic structure of the graph, we can extend the RNN analogy by remarking that the node-level score calculation combines inputs from multiple adjacent children nodes that are fed into an activation function (see Section 5.6 for a description of how the neuron parameters are set). In this model, each node is a neuron, and its children nodes are the neuron's dendrites. As with forward propagation in an RNN, our score propagation algorithm uses a breadth-first search that halts at nodes that have already been visited; once all nodes have been visited, we repeat the whole process and continue this repetition until the scores at the entry nodes have converged. The user can set several parameters to control the score propagation process as described in Algorithm 2's **Data** section.

An adversary generally wants to take the least risky path possible through the attack graph; consequently, the longer and more difficult the path, the less likely the adversary is to pursue it, and the lower the cumulative vulnerability score should be. In a similar fashion, we would consider an attack graph more vulnerable if there are multiple paths to the same attack outcome. To represent this idea in the cumulative vulnerability score calculation, we employ a probabilistic union function that gives the likelihood that an adversary will move into at least one of the attack paths flowing from a node's children [33]. This method is effective because it incorporates the children nodes' cumulative vulnerability scores (and the scores of their descendants, including attack outcomes) into the cumulative vulnerability scores of their parents. Consequently, the cumulative vulnerability score of an entry node represents the vulnerability of all attack pathways reachable from that location.

Algorithm 2: Propagating Cumulative Vulnerability Scores

Data: Attack graph G with nodes n_1, ... ,$n_k \in N$; entry nodes $A \subset N$; attack outcome nodes $L \subset N$; exponentially-weighted moving average parameter α; entry node cumulative score change parameter sum_Ratio; maximum number of repetitions parameter max_Cycles
Result: Attack graph G with cumulative vulnerability scores at each n_i
$average\_Sum = \mid A \mid$;
while $\frac{average\_Sum}{|A|} > sum\_Ratio$ and $count < max\_Cycle$ **do**
  $average\_Sum = 0$;
  Queue $Q \longleftarrow A$;
  while $\mid Q \mid > 0$ **do**
    $n \leftarrow Q$.dequeue;
    if n has not been visited **then**
      // c_i is a child of node n
      $union\_Probability = n.intrinsic\_Score \times (1 - \prod_{i=1}^{k}(1 - (c_i \times edge\_Defense(n, c_i))))$;
      /* The step below represents the activation function. See Section 5.6 to learn about the process used for parameter selection */
      $n.cumul\_Score = -7^{-union\_Probability} + 1$;
      if $n \in A$ **then**
        $average\_Sum = average\_Sum + |n.cumul\_Score - n.exp\_Avg|$;
      end
      $n.exp\_Avg = \alpha \times n.cumul\_Score + (1 - \alpha) \times n.exp\_Avg$;
      Mark n as visited;
      for $parent\_Node$ in $n.parents$ **do**
        if $parent\_Node$ has not been visited **then**
          Q.append($parent\_Node$);
        end
      end
    end
  end
  $count++$;
end

5.4 Optimizing Defense Placement

The optimization component of GRAVITAS adds defenses to the system with the goal of minimizing objective functions specified by the user. It does this by creating a "history" H containing the state of the attack graph after adding each successive defense. The ordering of the defenses in H is determined by a local objective function. The optimal defense set $\widetilde{M}$, which corresponds to a "moment" in history, $h \in H$, is ultimately decided by a global objective function. Algorithm 3 provides an overview of the optimization algorithm, whereas Algorithm 4 describes the **refresh_Defense_Set**(d,G) function that is used to generate the set of defenses to be compared by the local objective function. Section 5.4.1 describes the local and global objective functions in detail.

5.4.1 Objective Functions

Choose from all defense set members $s \in S$, where s consists of a just-added defense d and corresponding graph G.

$$\textbf{min}[\alpha\_Local \times s.d.cost + (1 - \alpha\_Local) \times adversary\_Score(s.G)] \quad (1)$$

Algorithm 3: Optimizing System

Data: Attack graph G; defenses d_1, ... ,$d_j \subset M$; maximum number of defenses to be added $max\_Defenses$; defense addition halting parameter opt_Halt_Value; global objective function parameter α_Global; local objective function parameter α_Local
Result: H, h_Best; /* h_Best includes the optimally-defended graph G and optimal defense set $\widetilde{M} \subset M$ */
S = **refresh_Defense_Set**($\emptyset$, G); /* S is a set containing graphs that each differ in only one defense */
$H = \emptyset$;
repeat
  $chosen\_Min\_Obj\_Val = \infty$;
  for s in S **do**
    $local\_Obj\_Value$ = **local_Objective_Func**(α_Local, $s.d.cost$, adversary_Score($s.G$));
    if $local\_Obj\_Value < chosen\_Min\_Obj\_Val$ **then**
      $chosen\_Set = s$;
    end
    $s.time\_In\_Set+ = 1$;
  end
  $total\_Cost = total\_Cost + chosen\_Set.d.cost$;
  H.add(make_History_Moment($chosen\_Set.d$, $chosen\_Set.G$));
  S.remove($chosen\_Set$);
  S = **refresh_Defense_Set**($chosen\_Set.d$, $chosen\_Set.G$);
until $\mid S \mid \leq 0$ and $\mid H \mid \geq max\_Defenses$ and **global_Objective_Func**(α_Global, total_Cost($h.\widetilde{M}$), adversary_Score($h.G$)) $\leq opt\_Halt\_Value$;
$min\_Global\_Obj\_Val = \infty$;
for $h \in H$ **do**
  $global\_Obj\_Val$ = **global_Objective_Func**(α_Global, total_Cost($h.\widetilde{M}$), adversary_Score($h.G$));
  if $global\_Obj\_Val < min\_Global\_Obj\_Val$ **then**
    $min\_Global\_Obj\_Val = global\_Obj\_Val$;
    $h\_Best = h$
  end
end

Choose from among among all moments in history $h \in H$, where h consists of defense set $\widetilde{M}$ and corresponding graph G with all $d \in \widetilde{M}$ added.

$$\begin{aligned} &\textbf{min}[\alpha\_Global \times total\_Cost(h.\widetilde{M}) \\ &+(1 - \alpha\_Global) \times adversary\_Score(h.G)] \end{aligned} \tag{2}$$

The purpose of each objective function is to minimize the system's total vulnerability (*adversary_Score*) while simultaneously minimizing the cost of the defenses needed to lower the vulnerability. We employ two separate objective functions: local and global (Eq. 1 and 2, respectively). The local objective function is applied to every defense-graph pair in the current defense set; the pair that minimizes the objective function is added to the defense history. The global objective function is employed *after* the algorithm has completed populating the defense history; it chooses the optimal "moment" (cumulative set of defenses) from the history. Each function employs a user-specified parameter α that tells the function how to weigh the vulnerability against the cost.

Algorithm 4: Refreshing Defense Set

Function refresh_Defense_Set(d,G):
  Data: Defense set S; defense d just selected by objective function, graph G with optimal defense just added, maximum number of cycles max_Set_Time that a defense can be in the set; the maximum size of the defense set set_Size; the list of defenses that are not in the set and have not yet been chosen for optimization $available\_Defense\_List$
  Result: Defense set S with old defenses removed and new defenses added
  for s in S **do**
    if $s.timeInSet > max\_Set\_Time$ **then**
      S.remove(s);
      $available\_Defense\_List$.add($s.defense$)
    end
  end
  for $G \in S.graphs$ **do**
    G.apply_Defense($chosen\_Defense$);
    G.propagate_Defense($chosen\_Defense$);
  end
  $max\_Vul\_Device\_Defenses$ = All unused defenses from the device possessing the entry node with the highest cumulative vulnerability score;
  while $\mid S \mid < set\_Size$ and $\mid available\_Defense\_List \mid > 0$ **do**
    if $\mid max\_Vul\_Device\_Defenses \mid > 0$ **then**
      d = random_Defense($max\_Vul\_Device\_Defenses$);
    else
      d = random_Defense($available\_Defense\_List$);
    end
    new_G = deep_Copy(G);
    new_G.**apply_Defense**(d);
    new_G.**propagate_Defense**(d); /* Similar to Section 5.3.2, except that the first nodes in the queue are the nodes upon which the new defense is placed */
    S.add(d, new_G);
    $available\_Defense\_List$.remove(d);
  end
  return S;

5.5 Adversary Models

The vulnerability of a given IoT system can be expressed using a function of its entry node cumulative vulnerability scores (see Eq. 3). These scores contain information about the ease of exploitation and appeal of attack outcomes in the rest of the graph (see Section 5.3.2 for more detail). Given a set of entry nodes A, an adversary would likely want to enter the system at its most vulnerable location(s). As a result, our optimization process should minimize the cumulative vulnerability scores of the k highest-scoring entry nodes.

$$adversary\_Score = \max_{A^* \subset A, \mid A^* \mid = k} \frac{\sum_{i=1}^{k} a_i.cumul\_Score}{k} \tag{3}$$

Using only the highest-scoring entry node is not recommended because the $adversary\_Score$ tends to "plateau" (bottom out) after only a few defenses are added. This

happens because there may be no additional defenses that substantially impact the highest-scoring nodes. For good performance (and to adequately protect all parts of a complex system), the user should specify a k that is at least equal to the logarithm of the total number of devices. The user can also adjust the vulnerability of an entry node by changing its *accessibility* score.

This approach assumes that the adversary has white-box knowledge about the system, and that we, the defenders, are knowledgeable enough about an adversary's motivations to confidently assign quantitative *impact* scores to attack outcomes. However, the adversary may not be fully knowledgeable about all the defenses that we have added to the system, and we may not fully understand the adversary's motivations. To account for this, the user can instruct GRAVITAS to randomly select attack outcome *impact* scores drawn from a distribution, or to randomly remove selected defenses during the optimization process. By adding additional "noise" to the model, the user can ensure that their IoT/CPS is prepared for all manner of adversaries and attacks.

5.6 Parameter Validation

The vulnerability scoring process contains several additions and modifications from those used in previous articles. In order to validate these modifications, we created a system called TASC (Testing for Autonomously-Generated Smart Cities) that generates quasi-random IoT/CPS. Controllable parameters include the number of devices, the relative number of different device categories/subcategories, the distribution of connection types between different devices, defense types and costs, and *impact* scores for attack outcomes. In theory, large systems generated with the same parameters but with a different random seed should have similar properties and broadly similar optimization curves. When optimized using the same adversary model and propagation/optimization parameters (such as α_Local), these systems should trace a similar adversary score vs. defenses-added curve, and should also possess a similar globally-optimal solution for a given α_Global.

One of the novel features of GRAVITAS involves its defense-addition optimization process. Adding a defense changes the intrinsic vulnerability score of the nodes and edges affected by it, which in turn influences the cumulative vulnerability scores at the entry nodes once re-propagation is completed. The current scoring set in Table 1 was chosen by generating several broadly similar systems using different seeds and employing different scoring sets in each system's optimization procedure to identify which set resulted in the most consistent results. We chose the activation function for the union probability function (see Section 5.3.2) in a similar manner, comparing several different functions (including exponential, power, and logistic functions) with several different numeric values for each. By creating TASC systems with the same parameters but different random seeds, we were able to determine which defense score set and activation function combination was most likely to produce consistent results.

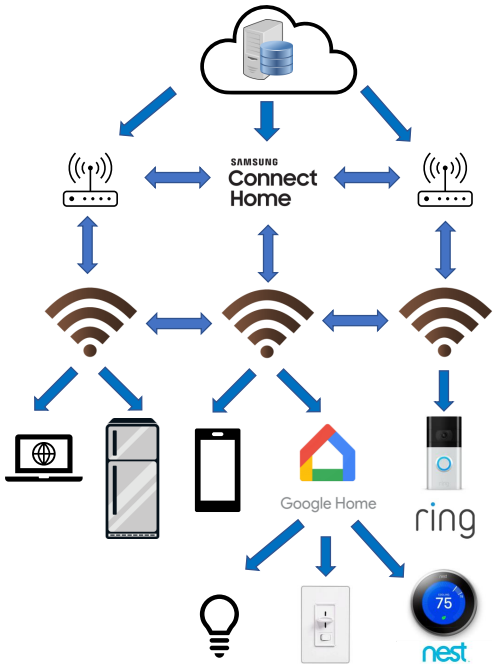


Fig. 7: A simplified representation of the "Smart Home" system. The real system contains 23 devices, including 3 WiFi routers, 4 local controllers, 11 sensors/actuators, and over 50 connections.

6 Example Application

To demonstrate the functionality of GRAVITAS, we created a sample "Smart Home" system involving common household devices. Fig. 7 shows a simplified version of the system used for this analysis. While real-world devices in this system would contain built-in defenses out-of-the-box, this system assumes that the devices initially contain no defenses so that GRAVITAS can add them. Fig. 8 shows the result of the optimization process using different values of α_Local (the local objective function weighting parameter) and k (the number of entry node cumulative vulnerability scores averaged to obtain the adversary score). Fig. 4 shows the complete attack graph for one of the intruder-detection sensors in the Nest system.

The graphs in Fig. 8 show that the adversary score curve generally becomes less "noisy" as k increases. This is expected given that we are averaging more scores for higher k. As α_Local increases, cost is weighted higher in the objective function, and thus the adversary score drops less quickly while the cost also increases less quickly. Changing k or the adversary score has a nonlinear impact on the optimal value (represented by the black line for $\alpha\_Global = 0.00024$).

In some of the graphs in Fig. 8, the adversary score appears to "plateau" for the first 20-30 defenses added. The plateau defies our intuition that the model should choose the "best" defenses early on. The uncharacteristically steep drop that usually follows a plateau indicates that the initial defense set contains few "useful" defenses; these defenses instead appear slightly later in the optimization process due to the inherent randomness of choosing defenses during initialization of the defense set. This can be mitigated by

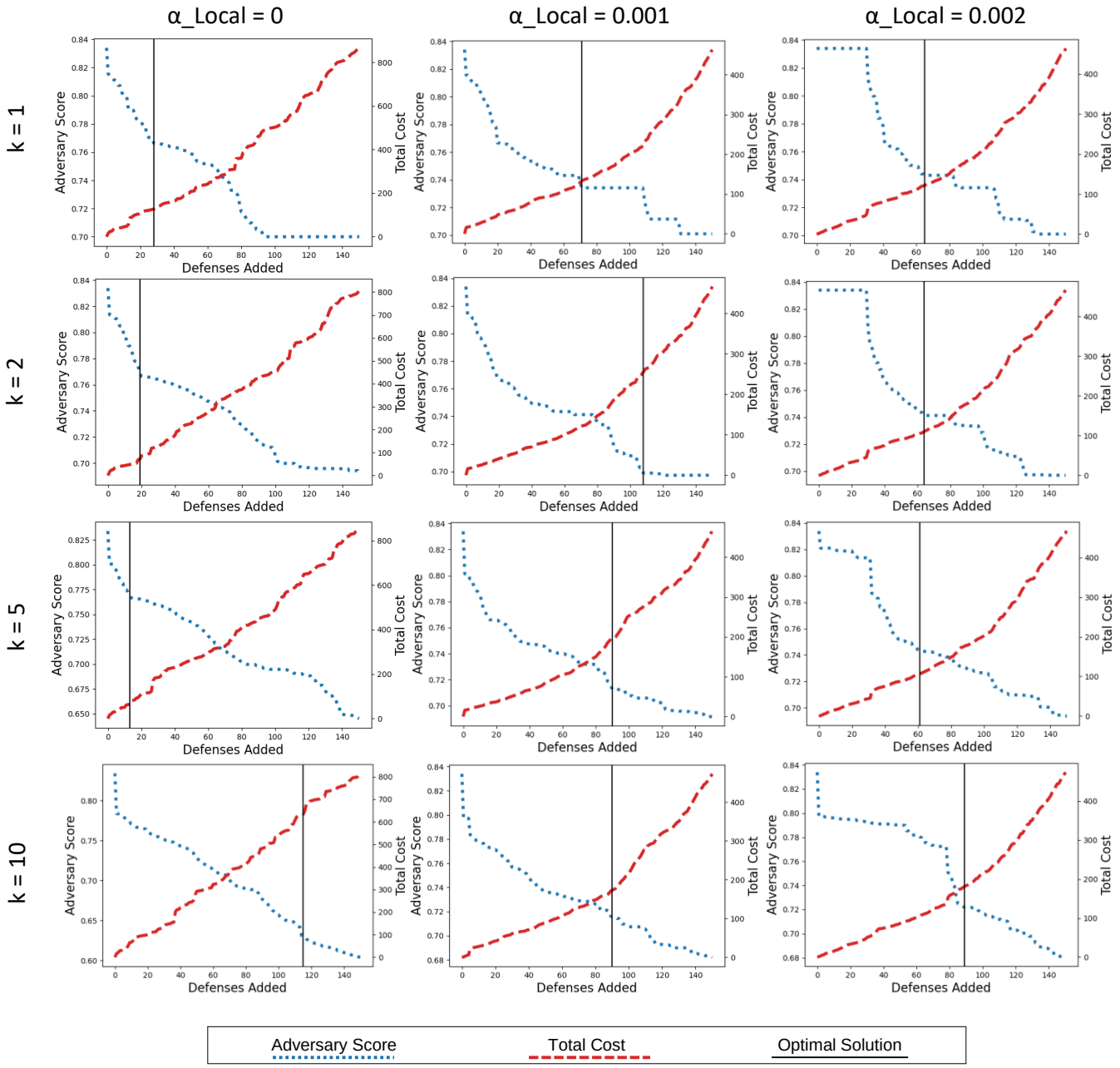


Fig. 8: Optimization curves and global optimal solutions during optimization of the "Smart Home" system. α_Local represents the parameter for the local objective function (higher α_Local means higher cost sensitivity), while k represents the number of entry nodes whose cumulative vulnerability scores are averaged to obtain the adversary score. The black line represents the global optimal solution for $\alpha\_Global = 0.00024$.

increasing the capacity of the defense set at the cost of a linear increase in run time. The inverse phenomenon (a lack of "useful" defenses) often causes a similar plateau near the end of the optimization process. This is because the remaining defenses have relatively little impact on the adversary score and/or a high cost, meaning that their contribution to lowering the objective function value is extremely limited.

7 Discussion

While GRAVITAS is a useful heuristic, it is not perfect. The categories and subcategories try to encompass the wide range of IoT/CPS devices that exist today, but they cannot realistically be expected to cover all devices. For example, the authentication procedures of embedded devices vary widely, not least because of the dozens of different security protocols and miniature operating systems that these devices use [34]. In addition, the limited graph templates of GRAVITAS may lead the user to "pigeon-hole" their device into an ill-fitting category, accidentally including attacks that may not be possible in real life and excluding security flaws that are yet to be discovered. Fortunately, GRAVITAS was built to be adaptable: users can add additional device templates easily, including those related to 5G networks,

which might provide insight into the debate many countries are having about the supposed threat of employing foreign-manufactured devices in their 5G networks. Other applications of GRAVITAS include the design of medical body-area networks, smart city systems, manufacturing facilities, and public utility networks.

Like any other model, GRAVITAS is unable to comprehensively represent all the software and hardware vulnerabilities unique to each device. However, tools such as CVE-Search can be used to extract an up-to-date list of vulnerabilities for every device, while off-the-shelf software (or the method described in Ghazo et al.) can be used to match these vulnerabilities with those in the template attack graphs [24], [26], [35], [36]. A future version of GRAVITAS could incorporate these real-world vulnerabilities, allowing us to find novel attack paths specific to each system. This would allow GRAVITAS to build a more accurate picture of an IoT system, resulting in better model performance.

The optimization component of GRAVITAS could also be improved. Its "greedy local search" methodology does not consider how adding a defense in the current round will affect the objective function value in later rounds. One possibility is to add "lookahead" functionality that simply adds two defenses instead of one to each new entrant in the defense set. A more complex approach would see the local objective function augmented to include information about past iterations of the attack graph, perhaps employing a nonlinear estimator for future defense additions.

Despite the cyclic nature of the attack graphs, there were never any convergence issues during the millions of cumulative vulnerability score propagation cycles we ran during parameter validation and the "Smart Home" tests. Virtually every propagation iteration completed in 50 cycles or less, and none exceeded 100 cycles of propagation. However, there were some issues regarding the consistency of vulnerability scores among different propagation cycles: the exact same graphs with the exact same defenses applied in a different order would sometimes have a slightly different maximum vulnerability score. This is likely due to rounding errors that compound after several thousand floating-point calculations during score propagation. While these errors were not substantial in the "Smart Home" system, they do point to the possibility of a "Butterfly Effect" phenomenon in larger systems, where a different random seed or slightly-modified system can have a larger impact on the outcome. To guard against this, the users should perform the entire optimization process multiple times with different seeds so that they have several defense histories from which to pick the optimal solution.

Another issue with GRAVITAS lies in its treatment of defenses. Adding new hardware defenses is difficult post-production, and although it is theoretically possible to add software updates to an existing device, this is not always feasible. For an extant IoT/CPS, rearranging the connections between devices is often far more feasible than changing the devices themselves. Future versions of GRAVITAS should not just be able to add node and edge defenses, but also rearrange the system topology, including permissions and local network connections.

8 Conclusion

GRAVITAS is a useful heuristic for a complex security problem. Though imperfect, GRAVITAS provides new insights into the vulnerabilities of complex IoT/CPS, suggesting new attack paths overlooked by trained experts and applying strategically-placed defenses that reduce the system vulnerability. It also provides a way to dramatically shorten the tedious pen-testing procedures that have so far characterized IoT/CPS security research. Most importantly, GRAVITAS allows for an organization to fix the design of an IoT/CPS before deployment, providing a proactive security solution that takes a holistic view of the system. In an era where IoT/CPS will soon be ubiquitous, getting the security right the first time is essential. As a security model specifically tailored to the unique devices and complex topology of IoT/CPS, GRAVITAS could become an important tool in the arsenal of security practitioners.

References

[1] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerging Topics in Computing*, vol. 5, no. 4, pp. 586–602, 2017.

[2] A. Bosche, D. Crawford, D. Jackson, M. Schallehn, and C. Schorling. (2018) Unlocking opportunities in the Internet of Things. Report. Bain Company. San Francisco, California. [Online]. Available: https://www.bain.com/contentassets/5aa3a678438846289af59f62e62a3456/bain_brief_unlocking_opportunities_in_the_internet_of_things.pdf

[3] A. O. Akmandor and N. K. Jha, "Smart health care: An edge-side computing perspective," *IEEE Consumer Electronics Magazine*, vol. 7, no. 1, pp. 29–37, 2018.

[4] B. L. R. Stojkoska and K. V. Trivodaliev, "A review of Internet of Things for smart home: Challenges and solutions," *J. Cleaner Production*, vol. 140, pp. 1454–1464, 2017.

[5] R. Zhang and X. Liu, "IoT-based maintenance process design for fusion reactor remote handling system," *J. Fusion Energy*, vol. 33, no. 6, pp. 653–657, 2014.

[6] M. Yun and B. Yuxin, "Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid," in *Proc. IEEE Int. Conf. Advances in Energy Engineering*, 2010, pp. 69–72.

[7] A. Al-Ali and R. Aburukba, "Role of Internet of Things in the smart grid technology," *J. Computer and Communications*, vol. 3, no. 05, p. 229, 2015.

[8] S. K. Datta, R. P. F. Da Costa, J. Härri, and C. Bonnet, "Integrating connected vehicles in Internet of Things ecosystems: Challenges and solutions," in *Proc. IEEE Int. Symp. A World of Wireless, Mobile and Multimedia Networks*, 2016, pp. 1–6.

[9] A. Thierer and A. Castillo. (2015) Projecting the growth and economic impact of the Internet of Things. Mercatus Center. Arlington, Virginia. [Online]. Available: https://www.mercatus.org/system/files/IoT-EP-v3.pdf

[10] J. Stavridis and D. Weinstein. (2016, 11) The Internet of Things is a cyberwar nightmare. Foreign Policy Magazine. [Online]. Available: https://foreignpolicy.com/2016/11/03/the-internet-of-things-is-a-cyber-war-nightmare/

[11] J. A. Lewis. (2016, 2) Managing risk for the Internet of Things. Center for Strategic International Studies. [Online]. Available: https://csis-website-prod.s3.amazonaws.com/s3fs-public/legacy_files/files/publication/160217_Lewis_ManagingRiskIoT_Web_Redated.pdf

[12] E. J. Markey and R. Blumenthal. (2020, 6) Letter to acting administrator James Owen concerning cybersecurity issues with Internet-Connected Cars. United States Senate. [Online]. Available: https://www.markey.senate.gov/imo/media/doc/NHTSA%20Cybersecurity%20Followup.pdf

[13] J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim, and J. N. Kim, "An in-depth analysis of the Mirai botnet," in *Proc. Int. Conf. Software Security and Assurance*, 2017, pp. 6–12.

[14] V. Sehwag and T. Saha, "TV-PUF: a fast lightweight analog physical unclonable function," in *Proc. IEEE Int. Symp. Nanoelectronic and Information Systems*, 2016, pp. 182–186.

[15] K. McKay, L. Bassham, M. Sönmez Turan, and N. Mouha, "Report on lightweight cryptography," National Institute of Standards and Technology, Tech. Rep., 2016.

[16] A. Mohsen Nia, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Physiological information leakage: A new frontier in health information security," *IEEE Trans. Emerging Topics in Computing*, vol. 4, no. 3, pp. 321–334, 2016.

[17] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho. (2011) Stuxnet under the microscope. ESET LLC. [Online]. Available: https://www.esetnod32.ru/company/viruslab/analytics/doc/Stuxnet_Under_the_Microscope.pdf

[18] G. Hernandez, O. Arias, D. Buentello, and Y. Jin. (2014) A smart Nest thermostat: A spy in your home. Black Hat USA. [Online]. Available: https://www.blackhat.com/docs/us-14/materials/us-14-Jin-Smart-Nest-Thermostat-A-Smart-Spy-In-Your-Home.pdf

[19] A. Juels, R. L. Rivest, and M. Szydlo, "The blocker tag: Selective blocking of RFID tags for consumer privacy," in *Proc. ACM Conf. Computer and Communications Security*, 2003, pp. 103–111.

[20] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. ACM Wkshp. Security and Artificial Intelligence*, 2011, pp. 43–58.

[21] IBM Security. (2019) Penetration testing: Protect critical assets using an attacker's mindset. IBM Corporation. [Online]. Available: https://www.ibm.com/security/services/penetration-testing

[22] A. Ur-Rehman, I. Gondal, J. Kamruzzuman, and A. Jolfaei, "Vulnerability modelling for hybrid IT systems," in *Proc. IEEE Int. Conf. Industrial Technology*, 2019, pp. 1186–1191.

[23] X. Ou, S. Govindavajhala, and A. W. Andrew, "MulVal: A logic-based network security analysis," in *Proc. Usenix Security Symposium*, 2005.

[24] S. Jajodia and S. Noel, *Topological Vulnerability Analysis*. Center for Secure Information Systems, George Mason University, 2009, pp. 139–154.

[25] M. Malowidzki, D. Hermanowski, and P. Berezinkki, "Tag: Topological attack graph analysis tool," in *Proc. Cyber Security in Networking Conference*, 2019, pp. 158–160.

[26] A. T. Al Ghazo, M. Ibrahim, H. Ren, and R. Kumar, "A2G2V: Automatic attack graph generation and visualization and its applications to computer and SCADA networks," *IEEE Trans. Systems, Man, and Cybernetics: Systems*, pp. 1–11, 2019.

[27] M. Barrère and E. C. Lupu, "Naggen: A network attack graph generation tool - IEEE CNS 17 poster," in *Proc. IEEE Conf. Communications and Network Security*, 2017, pp. 378–379.

[28] T. Saha, N. Aaraj, N. Ajjarapu, and N. K. Jha, "SHARKS: Smart Hacking Approaches for RisK Scanning in Internet of Things and Cyber Physical Systems based on Machine Learning," under review, 2020.

[29] FIRST. (2019) Common vulnerability scoring system version 3.1 specification document. FIRST Inc. [Online]. Available: https://www.first.org/cvss/v3-1/cvss-v31-user-guide_r1.pdf

[30] B. Kordy, L. Pièvre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *Computer Science Review*, vol. 13-14, pp. 1 – 38, 2014.

[31] X. Ou and A. Singhal, *Quantiative Security Risk Assessment of Enterprise Networks*. New York, NY: Springer, 2012.

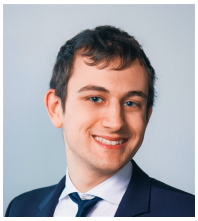
[32] S. Noel, L. Wang, A. Singhal, and S. Jajodia, "Measuring security risk of networks using attack graphs," *Int. J. Next Generation Computing*, vol. 1, Jan. 2010.

[33] M. U. Aksu, M. H. Dilek, E. İ. Tatlı, K. Bicakci, H. I. Dirik, M. U. Demirezen, and T. Aykır, "A quantitative CVSS-based cyber security risk assessment methodology for IT systems," in *Proc. IEEE Int. Carnahan Conf. on Security Technology*, 2017, pp. 1–8.

[34] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, "Operating systems for low-end devices in the Internet of Things: A survey," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 720–734, 2016.

[35] A. Dulaunoy, P.-J. Moreels, and R. Vinot. (2020) CVE-search project. CVE-Search. [Online]. Available: http://cve-search.org

[36] OpenVAS - Open Vulnerability Assessment Scanner. Greenbone Networks GmbH. [Online]. Available: https://www.openvas.org/



Jacob Brown Jacob Brown received his Bachelor of Science in Electrical Engineering (Hons.) from Princeton University. He has conducted machine learning (ML) research for security and healthcare applications, such as creating a COVID-19 outbreak prediction model based on micro-level demographic data. Jacob has also worked on ML-based geospatial modeling of conflict zones. A member of Sigma Xi, he has received several grants and prizes in recognition of his work, including a grant from the Project X Innovation Fund and the G. David Forney Jr. Prize for an outstanding record in communication sciences.

Tanujay Saha Tanujay Saha is currently pursuing his Ph.D. degree at Princeton University, NJ, USA. He received his Bachelors in Technology (Hons.) in Electronics and Electrical Communications Engineering from Indian Institute of Technology, Kharagpur, India in 2017. He has held research positions in various organizations and institutes like Intel Corp., KU Leuven (Belgium), and Indian Statistical Institute. His research interests lie at the intersection of IoT, cybersecurity, machine learning, embedded systems, and cryptography. He has both industry and academic experience in the theoretical and practical aspects of cryptography and machine learning.

Niraj K. Jha Niraj K. Jha received the B.Tech. degree in electronics and electrical communication engineering from I.I.T., Kharagpur, India, in 1981, and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign, Illinois, in 1985. He has been a faculty member of the Department of Electrical Engineering, Princeton University, since 1987. He was given the Distinguished Alumnus Award by I.I.T., Kharagpur. He has also received the Princeton Graduate Mentoring Award. He has served as the editor-in-chief of the IEEE Transactions on VLSI Systems and as an associate editor of several other journals. He has co-authored five books that are widely used. His research has won 20 best paper awards or nominations. His research interests include smart healthcare, cybersecurity, machine learning, and monolithic 3D IC design. He has given several keynote speeches in the area of nanoelectronic design/test and smart healthcare. He is a fellow of the IEEE and ACM.