How Secure is Your IoT Network?

Joshua Payne* Karan K. Budhraja† Ashish Kundu‡
*Stanford University, CA, USA Email: joshp007@stanford.edu
†University of Maryland, Baltimore County, MD, USA Email: karanb1@umbc.edu
‡IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA Email: akundu@us.ibm.com

***Abstract*—The proliferation of IoT Devices is wide-spread and continuing to increase in a superlinear manner. As a result, the complexity and commonality of cybersecurity challenges in new and dynamic environments are rapidly increasing. The following question arises: how can an IoT network administrator determine the network's vulnerabilities, as well as the extent of the network's holistic security risk, especially given the increasing complexity of IoT networks? In this paper, we propose and implement the notion of an *attack circuit*: a system and set of techniques for the holistic assessment of cybersecurity risks for a network of IoT devices. Our system provides a practical method of finding answers to the aforementioned questions and gives insight into the possible attack paths an adversary may utilize. We also propose the use of two types of risk measures, *compositional scores* and *dynamic activity metrics*, which are used to evaluate IoT devices and the overall IoT network in the context of an attack circuit, and demonstrate the effectiveness of attack circuits as practical tools for computing these scores as well as finding optimal attack paths (by metrics of exploitability, impact, and risk to confidentiality, integrity, and availability) in heterogeneous, extensible IoT networks.**

I. Introduction

Statista estimates that the number of connected IoT devices will rise from 19.4B in 2018 to 34.2B in 2025 [1], and that the percentage of U.S. homes that are "smart" will rise from 33.2% in 2019 to 53.9% in 2023 [2]. Usage of Internet of Things (IoT) devices in networks such as smart homes, smart cities, and digital healthcare is clearly increasing, and while the adaptable and heterogeneous features of these networks have proven to be crucial in solving a vast array of different issues, they have also given adversaries a veritable sandbox of vulnerabilities to exploit. Security measures that prevent attackers from exploiting these vulnerabilities are more important now than ever before because of the growing quantity and sensitivity of data that people are putting online. The complexity multi-stage privacy, service, and safety attacks is also growing, and it is now critical for home owners, enterprises, or government organizations that host various IoT devices on complex IoT networks (augmented with the Bring Your Own Device (BYOD) trend of devices) to be aware of the cybersecurity risks that may be present. In particular, it is vital to determine high-priority vulnerabilities that need to be addressed in order to keep the overall network, as well as the individual devices, protected.

We propose the notion of an *attack circuit*, a structure that arises very naturally from the often extensible, modular, and heterogeneous nature of IoT networks and helps to model possible attack paths and evaluate the security state of the represented IoT network as well as each individual device therein. Current methods, discussed in Section II, seek to address these problems as well for general networks as well as specifically for IoT networks–our work builds off of these, bringing ideas from this work together as well as adding our own novelties to address some of the limitations of current work. These novelties solve problems such as vulnerability documentation processing, delivering different types of metrics in the form of *compositional scoring* and network flow analysis, and incorporating network activity data into *dynamic activity metrics* (this is explored further in Section III). In Section IV, we discuss the practical and theoretical usefulness of this notion in the context of a smart home, using vulnerability data from the National Vulnerability Database (NVD)[1], network traffic data from off-the-shelf IoT devices, and optimization and machine learning techniques for constructing and evaluating the resulting attack circuits and attack paths. Finally, in Section VI, we demonstrate our own implementation of the attack circuit and evaluate the effectiveness of such a notion through quantitative experimentation. This is followed by concluding remarks in Section VII.

II. Related Work

In computer security, a *vulnerability* is defined as a weakness of a system that can be exploited by an attacker. The attacker may then perform unauthorized activities within the system. Alternatively, an *exposure* is a software error in the system that allows the attacker to gain access to system data and conduct information gathering activities. The attacker may accompany this by hiding unauthorized system activity from associated monitoring services. Subsequently, the Common Vulnerabilities and Exposures (CVE) system [3] is a built reference for publicly identified information-security vulnerabilities and exposures. The system is maintained by the Mitre Corporation[2]. CVE entries are primarily composed of identifiers, descriptions, references and the date at which the CVE entry was created. The Mitre Corporation also maintains the Common Weakness Enumeration (CWE) system [4], which categorizes software weaknesses and vulnerabilities. The combined use to CVE and CWE allows organizations to select appropriate software tools for internal usage. Our work uses the CVE and CWE systems as a standardized

[1]http://nvd. nist.gov/download.cfm
[2]mitre.org

source of information to generate a representation of possible attacks. Additionally, the Common Vulnerability Scoring System (CVSS) [3] provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity. We build on these existing scores to evaluate device and network vulnerability.

Work in [5] discusses the threat to user privacy in a smart-home environment, using multi-stage privacy attacks. The attacks are evaluated to be effective in determining the state and actions of the devices in both unencrypted and encrypted communication settings. The work is evaluated using common IoT devices and multiple attack protocols (passive, observer attack). With the use of a large degree of automation in activity detection and identification (using machine learning techniques and network traffic data mining), this work exemplifies the security threats faced by smart-home networks. Work in [6] also emphasizes the security risks of smart-home networks using commercially available smart-home devices with encrypted communication. The authors explore attacks by first identifying the device (using Domain Name System (DNS) queries or device fingerprinting) and then inference of activities based on changes in network traffic. Network traffic-based threats are also highlighted in [7], where the authors demonstrate an attacker that passively observes encrypted network traffic to infer sensitive details about network users.

Work in [8] examines the security flaws for smart-home networks with specific interest in the exploitation of the lack of mechanisms for firmware updates or patches for security vulnerabilities. The authors propose a system to identify the types of devices that are connected and suggest the use of appropriate communication constraints, given that knowledge. The device type for their work is the enumeration of a specific device. Work in [8] is, however, limited to the formulation of a method to identify a given device, and does not provide a metric to determine how vulnerable a device is. Network attacks are modeled using attack graphs in [9]. The authors propose a scalable model zero-day exploits [10] and client-side attacks [11], [12], in contrast to prior attack graph systems that focus on server-side vulnerabilities [13], [14]. The work, however, focuses on modeling such attacks and corresponding countermeasures, but does not provide a means to quantify the relative impacts of different attacks. Work in [15] discusses the provision of security objectives for smart-home networks. While it does not discuss the underlying detection mechanism used, the application of security flaw detection is aligned with the motivation for our work.

Term Frequency-Inverse Document Frequency (TF-IDF) [16] is a numerical statistic used in information retrieval to determine the relative importance of words in a document. TF-IDF and TF and IDF individually may also serve as heuristics for weighting words. Our work leverages TF-IDF to compute attack meta-data for a given CVE entry. TextRank [17], [18] is a graph-based ranking model for text processing. It is inspired by recursive graph-based ranking algorithms such as HITS [19] and PageRank [20], using a voting mechanism. While TextRank may be used for sentence extraction and text summarization, our work uses the information stored in the intermediate process: the extraction and ranking of phrases.

III. Problem Definition

Given a network of IoT devices and any additional knowledge about them (e.g., from CVEs or the device specification), the problem addressed by our work is to compute a security state triple $\langle R, E, I \rangle$ corresponding to the risk triple, exploitability score, and impact scores for each vulnerability, device, and the network. Exploitability is a measure of how difficult it would be for an adversary to compromise the object, and the impact is a measure of the level of harm or compromise an adversary could inflict in the case of vulnerability exploitation. Risk is meant to be interpreted as a holistic measure of the security state of the CVE, device, or network, which evaluates the confidentiality, integrity, and availability risks of the object's potential vulnerabilities: $R = \langle R_{Conf}, R_{Integ}, R_{Avail} \rangle$. R_{Conf} measures the impact of a successfully exploited vulnerability on the confidentiality of information managed by the device or network. A value of *Low* for R_{Conf} means that there is a low risk of disclosure of such information to unauthorized individuals or systems. R_{Integ} measures the impact of a successfully exploited vulnerability on the integrity of the system. For instance, if R_{Integ} has a value of *Complete*, an unauthorized user may be able to easily gain root access to a device following the exploitation of an associated vulnerability. R_{Avail} measures the impact of a successfully exploited vulnerability on the availability of the devices or networked services involved. This may include disk space, bandwidth/latency, and the uptime of the devices and components involved. Having a high availability risk would be particularly alarming for medical IoT networks, where lives depend on device fidelity and responsiveness.

The problem also incorporates the generation of a representation of the scored system for further assessment, including identifying possible attack paths that an adversary may traverse to carry out multi-stage attacks on the network. This may include network visuals, analysis-ready representations, lists of likely attack paths with respect to different metrics, and flow network problem solutions for downstream score computation. Such a representation would provide insights into otherwise very complex and unique IoT networks, and would give improved information for our security state triple.

Because vulnerabilities often stem from the way a system is used, the problem additionally incorporates data learned from the traffic of a particular device and holistic network behaviors. For instance, anomalous traffic volume can often be an indication of Denial-of-Service (DoS) attacks, and instances when a device is suddenly receiving responses from or sending requests to a blacklisted IP address could factor into our security state triple. This broadens the scope of our security state assessment and could provide strategies for real-time countermeasures against adversaries.

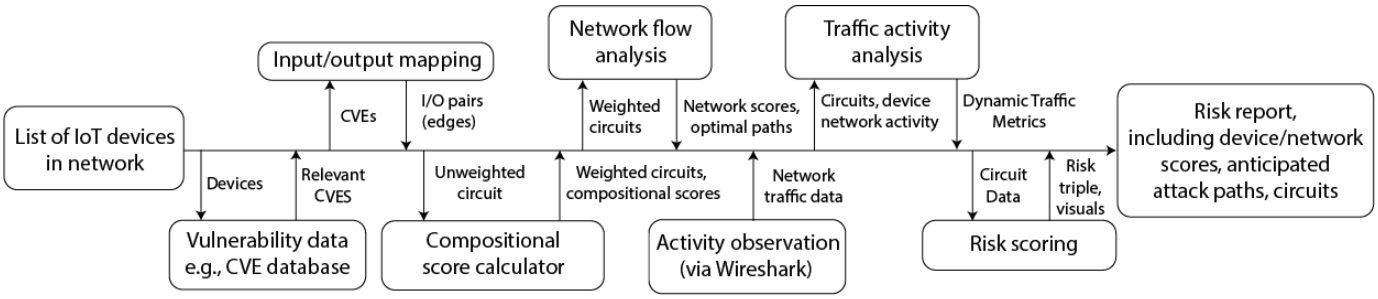

Fig. 1. System architecture summary.

IV. Proposed Method

Our proposal is a sequential computation of the triple $\langle R, E, I\rangle$—a method that utilizes preexisting knowledge about the devices in the network, topology of the network (determined by the relation between different vulnerabilities and potential network flow between devices), and dynamic device activity and network traffic information. This process is outlined in Figure 1. At the practical level, the attack circuit model we are proposing considers each of these properties and can provide each of the desired metrics. An attack circuit is a type of flow network [21], wherein the flows can be used to evaluate level of risk, exploitability, and impact of a network, device, and vulnerability. In this section, we will discuss the nontrivial problem of constructing attack circuits using preexisting knowledge and inferences about IoT devices in the network. We will then explore how the network can then be used to compute *compositional scores* on each of the vulnerabilities, which provide an important baseline for understanding the initial risk, exploitability, and impact assessments of the network. Then, we'll examine the affect that dynamic network traffic behaviors have on the security of devices in the network and show how these *dynamic metrics* can be combined with the compositional score for a more holistic look at the security of the IoT network. Finally, we'll propose the use of network flow algorithms that make use of the scores we've calculated for anticipating optimal attack paths, which give a final evaluation of the risk, exploitability, and impact measures of the network and its devices. A summary of the implemented system architecture is shown in Figure 1.

A. Circuit Construction

The attack circuit is constructed in two stages, described in the following sections.

1) Input/Output Extraction: The attack circuits are modeled using text input (vulnerability descriptions) from a vulnerability database. For each item in the database, a corresponding *input/output* pair is generated. The *input* corresponds to the attack source and the *output* corresponds to the attack target. The process is based on TF-IDF [16] and TextRank [17], [18] heuristics and is described next as a series of steps.

All text is primed by conversion to lowercase, removal of non-alphanumeric characters, tokenization, stemming [22], [23] and subsequent de-tokenization. TF-IDF is then used on the processed corpus to produce an ordering of tokens for each description. TextRank is used on the processed corpus to produce an ordered list of candidate phrases (with Part-Of-Speech (POS) tags) that may best represent the description. The ordered list is filtered to remove noun items (NN). For each item in the list, tokens pruned (limited to a maximum quantity of 3) based on TF-IDF ordering. Stemming is then removed from tokens by matching them to the corresponding phrase. The result is stored as *input*. This extraction process is then repeated using filtering to remove non-noun items and the result is stored as *output*.

2) Graph Composition: After all of the *input/output* pairs are created, we have the information we need to build the attack circuit structure. In our methodology, an attack circuit is a directed graph isomorphic to a flow network $C = (D, A, S, E)$ where $d \in D$ is a device, represented as a set of vertices that are the set of vulnerability database entries corresponding to d; A is the set of attacker vertices; S is the set of target (or sink, as we'll see later) vertices which represent the attack targets in the network; and E is the set of labeled, directed edges. The attack circuit scheme suffices to provide a logical approach to determining which attack targets are at risk. We are more interested, however, in creating variants of the attack circuit that may give insight to the potential impact, exploitability, and overall risk an IoT network yields. To do this, we assign weights and capacities to each of the edges depending on the impact and expoitability scores associated with the respective vulnerability data. A device's impact, exploitability and base scores are then used to compute the attack vector corresponding to its vulnerability. This is used as a component of the device's risk score.

B. Network Composition Analysis

We're interested in taking a holistic look at devices when evaluating their levels of security and risk. We therefore use several different security metrics that may each be classified into one of two larger categories: compositional scoring and dynamic activity assessment, discussed in the next subsection. Compositional scoring may be performed by observing the device specification, associated vulnerability database information and their corresponding vulnerability scores, and attack circuit topology. Our work incorporates the first two items

and designates the third for future work. These compositional scores are derived from the devices themselves and their role in the network composition, and can be calculated irrespective of ways the devices are being used over time. For our work, device's vulnerabilities are scored using the risk base, exploitability, and impact subscore method provided by the existing CVSS v3 standard which accompanies most CVEs. After computing the exploitability and impact subscores for each vulnerability, we can then calculate the compositional score (c) of each vulnerability, which is recursively computed using the impact weights and exploitability capacities of the edges of the graph. This is summarized in Equation 1 and Equation 2, where C_i is the set of attack circuit input vertices (c_i), where there exists an edge (c_i, c), C_o is the set of attack circuit output vertices (c_o), where there exists an edge (c, c_o), and v_d is a dampening constant. The impact and exploitability subscores of each vulnerability of a particular device are then assimilated and transformed (using sigmoidal activation) to determine the impact and exploitability subscores.

$$c_{Exploitability} \mathrel{+}= v_d \sum_{c_i} c_{i_{Exploitability}} \tag{1}$$

$$c_{Impact} \mathrel{+}= v_d \sum_{c_o} c_{o_{Impact}} \tag{2}$$

C. Network Traffic Analysis

We next examine the dynamic activity metric of a given device in the network. A large body of work is focused on abnormality detection and scoring in network traffic patterns [5], [6], [7]. A vast variety of metrics may be used for results that shed light onto particular aspects of IoT device and network security. Our work focuses on ascertaining a small number of these metrics from packet sniffing on the network of IoT devices over a period time. The data collected includes device traffic activity, packet content encryption, and the source/destination information of the packets.

To illustrate our use of device traffic activity information, consider the example shown in Figure 2. Devices that are online for the majority of the time (i.e. Google Home Mini, Roku Media Player, HP Printer, and Belkin WeMo) are assigned larger respective multipliers to their exploitability subscore, since their connection leaves them more open to attack by an adversary. The Amazon Echo Dot, in this case, does not have as significant a multiplier applied to its exploitability subscore, since it is not online as frequently.

Next, we analyze the percentage of the packets that are sent from and received by each device and their usage of secure encryption protocols. We also check whether any source or destination IP is listed in an IP blacklist database. The encryption metric is used as a multiplier for the exploitability subscore, and the blacklisted IP metric is used as a multiplier for the impact subscore. The calculated the compositional scores and dynamic activity metrics for a device can then be used to improve network analysis and security risk scoring.

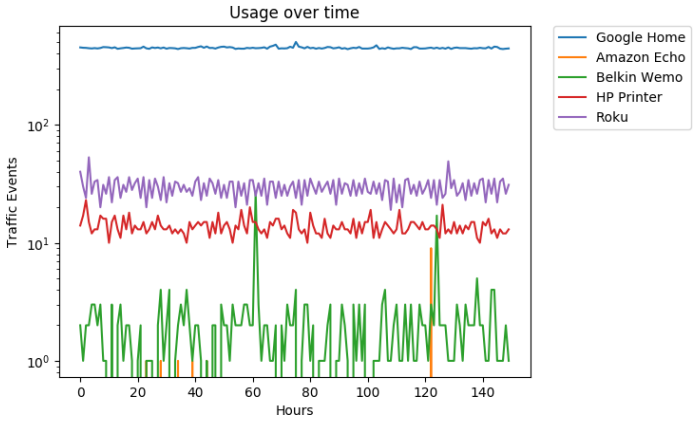

Fig. 2. An account of device traffic over a period of 12 hours.

D. Network Flows and Attack Path Analysis

To anticipate how an adversary might carry out an attack on the network and to score the network holistically, we apply a variety of *Network Flow Problems* [24] [25] to the attack circuit for evaluating potential attack paths based on impact, exploitability, and risk. Sources and sinks in the flow network correspond to attack sources and attack targets, respectively.

1) Impact Paths: An impact path is the route through the circuit that an attacker takes to maximize impact, defined by the circuit edge weights. We specify the attacker nodes as sources, and attacker targets as sinks. Equation 3 illustrates the *Maximum Flow Problem* method used–f_{uv} is the flow between vertices u and v, a is an attacker vertex, s is a sink (target) vertex. The sum of the impact path flows of an attack circuit (or the impact score of that circuit) are equivalent to the sum of the total impact of all *easily* accessible attack targets, where the degree of accessibility is defined by the exploitability of the attack paths leading to it.

$$\text{maximize} \quad \sum f_{as} \text{ subject to } \sum_j f_{ji} = \sum_j f_{ij} \tag{3}$$

2) Exploitability Paths: An exploitability path is a route from attacker to the attack target that is associated with a score denoting the *resistance* (intuitively, inverse exploitability) of that path. To determine the optimal exploitability paths in a circuit, we solve a *Minimum Cost Flow Problem*, where the cost of an edge is its resistance, or the inverse of the exploitability of that edge (e.g. (1−Exploitability), if Exploitability $\in [0, 1]$). We use the same sources and sinks as in Equation 3 now with a cost c_{ij} associated with edge (i, j) and a required flow r_{as} from attacker to sink (see Equation 4).

$$\text{minimize} \quad \sum c_{ij} f_{ij} \text{ subject to } \sum f_{as} = r_{as} \tag{4}$$

After computing the optimal exploitability and impact paths, the paths themselves may serve as information for network

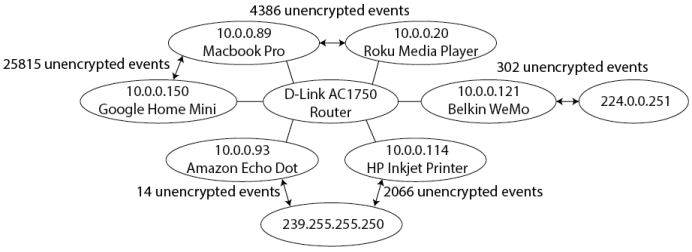

Fig. 3. The smart-home network traffic graph corresponding to Figure 2.

```
"Roku Media Player": [{
 "description": "The External Control
API in Roku and Roku TV products allow
unauthorized access via a DNS Rebind
attack.",
 "id": "CVE-2018-11314",
 "i/o": ["DNS Rebinding->this:Root
 Priv", "DNS Rebinding->this:Config
 File"] }]
```

Fig. 4. An example of a processed CVE data entry.

operators. We may also sum over the exploitability paths in the network to determine an overall network exploitability score, which we then combine with the impact score to improve an overall network security risk score.

3) Risk Flow Evaluation: Finally, we want to calculate high-risk paths and quantitatively apply these findings to our risk triple. A risk path is a route from attacker to target that is optimized for exploitability constraints, impact weights, and base compositional risk of the vertices. We combine the strategies used in exploitability path and impact path analysis here, solving a *Minimum Cost Maximum Flow Problem* to identify likely paths through the IoT network that an adversary might use in an attack. In Equation 5, we outline the network flow problem used to calculate risk, with the usual c_{ij} denoting cost of edge (i, j), a being an attacker vertex (source), and s being a target vertex (sink):

$$\begin{aligned} &\text{minimize } \sum c_{ij} f_{ij} \text{ subject to:} \\ &\text{maximize } \sum f_{as} \text{ subject to } \sum_j f_{ji} = \sum_j f_{ij} \end{aligned} \tag{5}$$

Computing risk paths allow us to calculate an improved risk triple for the network as well as the devices within. The triple $R = \langle R_{Conf}, R_{Integ}, R_{Avail} \rangle$ for a device is calculated using the CVSS v3 metrics for Confidentiality Impact, Integrity Impact, and Availability Impact associated with the CVEs of the device. Flow through each CVE is multiplied by the respective impact metric, and the result defines the risk triple R. The numerical values of the impact metrics may be found in table 8.4 of the CVSS v3 specification document[3].

V. Implementation

A. Our System

To understand the performance of our proposed method in common, real-world networks, we model our implementation with off-the-shelf smart home IoT devices. With our implementation based on the NVD, we narrow our focus to the 34 devices that are both common and have corresponding vulnerabilities listed in the NVD. Live experiments are run on a network consisting of five of these devices: an Amazon Echo Dot, a Belkin WeMo, an HP Inkjet Envy printer, a Google Home Mini, and a Roku digital media player (see Figure 3).

Our implementation is based on the CVE vulnerability database. Circuit construction is implemented using relevant CVEs in JSON format from NVD's database, with text processing using the Natural Language Toolkit (NLTK) [26]. An example of the CVE processed as in Section IV-A1 is shown in Figure 4. In this example, an *input* (left of the arrow in the i/o field) is an action (DNS rebinding attack) that the attacker needs to take to leverage the corresponding vulnerability, in this case, CVE-2018-11314 (for access to root privileges or the configuration file). For each input, there is an *output* (right of the arrow in the i/o field), which indicates the target that the attacker receives when once the input is applied to the vulnerability. A device may have multiple corresponding CVEs, a CVE may have multiple corresponding inputs, and an input may have multiple corresponding outputs. NetworkX [27] and SNAP [28] are then used to build the attack circuit using these *input/output* pairs and proceed with scoring.

B. Our Scoring Method

Let $EB_c \in [0, 10]$ denote CVE c's base exploitability score and $IB_c \in [0, 10]$ denote CVE c's base impact score. $EB_c, IB_c \in [0, 10]$. We use impact and exploitability scoring guidelines set by the NVD in their CVSS v3 method[4]. The numerical possibilities for the CVSS metrics AV, AC, PR, UI, I_{Conf}, I_{Integ}, and I_{Avail} can be found in Table 8.4 of the CVSS specification document. Scoring during the circuit construction phase is computed as shown in Equation 6. Numerical constants used are assigned based on empirical evaluation of the system.

$$\begin{aligned} &EB_c = 8.22 \times AV \times AC \times PR \times UI \\ &ISC_{Base} = 1 - [(1 - I_{Conf}) \times (1 - I_{Integ}) \times (1 - I_{Avail})] \\ &\texttt{if Scope = unchanged}: IB_c = 6.42 \times ISC_{Base} \\ &\texttt{else}: \quad IB_c = 7.52 \times [ISC_{Base} - 0.029] \\ &\qquad\qquad -3.25 \times [ISC_{Base} - 0.02]^{15} \end{aligned} \tag{6}$$

Then let EC_d denote device d's compositional exploitability score and IC_d denote device d's compositional impact score. Let c_i denote the set of CVEs with input to c (where each device d has a set of c corresponding to each CVE), let c_o denote the set of CVEs that c has output to, and let v_d denote

[3]https://www.first.org/cvss/specification-document

[4]http://nvd.nist.gov/cvss.cfm

a dampener variable (we used v_d = 0.1). Scoring during the compositional phase is computed as shown in Equation 7.

$$EC_d = \sum_{c \in d}(EB_c + v_d \sum_{i \in c_i} EB_i)$$
$$IC_d = \sum_{c \in d}(IB_c + v_d \sum_{o \in c_o} IB_o) \quad (7)$$

Next we compute the network traffic multipliers for impact and exploitability. Network traffic data is collected using Wireshark [29] and is stored in .pcap format. This file is then converted to CSV for parsing. The data is specific to a network of 5 IoT devices and a period of 4 days. We use three metrics: device Network Uptime (NU), Encryption Scheme (EN), and whether IP sources or destinations were listed in an IP blacklist database[5] (IP). The weights we assigned to the different categories correspond to importance with respect to the score in consideration—for instance, when considering the exploitability score, the Network Uptime multiplier = NU = {"always_online": 1.6, "frequently_online": 1.4, "rarely_online": 1.07, "never_online": 1}.

The final exploitability and impact scores E_d, I_d for a device d are then calculated as shown in Equation 8, where m_{E_i} is an exploitability-related network traffic multiplier (e.g. NU, EN), m_{I_j} is an impact-related network traffic multiplier (e.g. IP), and v_n is a normalizing variable (we use $v_{n_1} = v_{n_2} = 100$). These scores are normalized to a range of $[0, 1]$.

$$E_d = \tanh\left(EC_d \times \prod_i m_{E_i} \times v_{n_1}\right)$$
$$I_d = \tanh\left(IC_d \times \prod_j m_{I_j} \times v_{n_2}\right) \quad (8)$$

The overall network exploitability score E_N is defined by the sum of the exploitability scores of its devices, and is accompanied with the path of minimum cost to each of the attack targets. The overall network impact score I_N is the solution to the max-flow problem in the attack circuit after each of the edges have been weighted based on all of the CVEs' base impact scores. Finally, the risk triple $R_N = \langle R_{Conf}, R_{Integ}, R_{Avail} \rangle$ for the network is computed using the CVSS v3 metrics associated with the respective risk triples of the network's devices. First we solve the Maximum Flow Minimum Cost Problem in the attack circuit (see Equation 9).

$$\text{minimize} \sum EB_{ij} f_{ij} \text{ subject to:}$$
$$\text{maximize} \sum IC_{as} \text{ subject to } \sum_j f_{ji} = \sum_j f_{ij} \quad (9)$$

Once this is completed, the flow through each CVE is multiplied by the respective impact metric—Confidentiality Impact (R_C), Integrity Impact (R_I), and Availability Impact (R_A), as well as a normalization variable v_{n_i}, and the result

[5]https://myip.ms/browse/blacklist

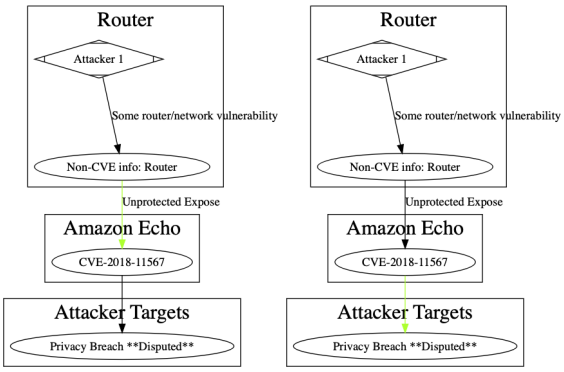

Fig. 5. Exploitability (left) and impact (right) circuits corresponding to one device under the observation of a single vulnerability.

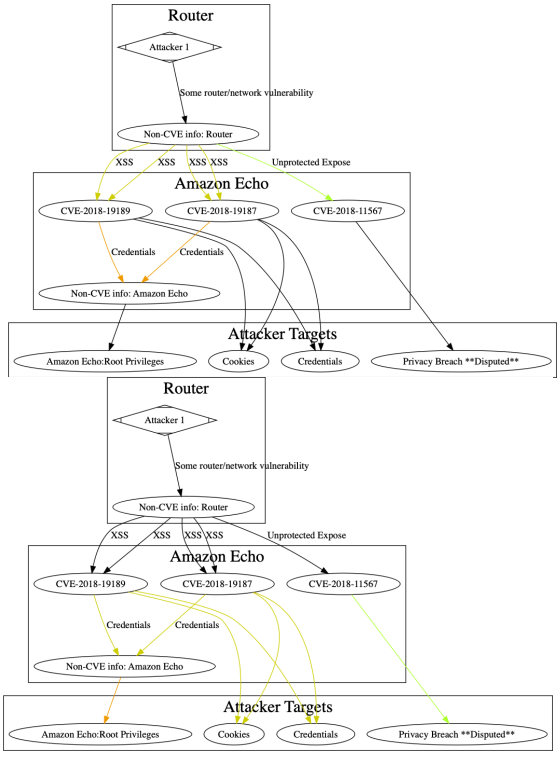

Fig. 6. Exploitability (top) and impact (bottom) circuits corresponding to one device under the observation of all vulnerabilities.

defines the risk triple R_d for a device d with CVEs c_i (shown in Equation 10). These scores are normalized to range $[0, 1]$.

$$R_{Conf} = \tanh\left(v_{n_1} \times R_C \sum_i f_{c_i}\right)$$
$$R_{Integ} = \tanh\left(v_{n_2} \times R_I \sum_i f_{c_i}\right) \quad (10)$$
$$R_{Avail} = \tanh\left(v_{n_3} \times R_A \sum_i f_{c_i}\right)$$

VI. Evaluation

We evaluate the attack circuit system using a variety of networks and device activity metrics. In our work, we showcase three of these networks: one consisting of one device

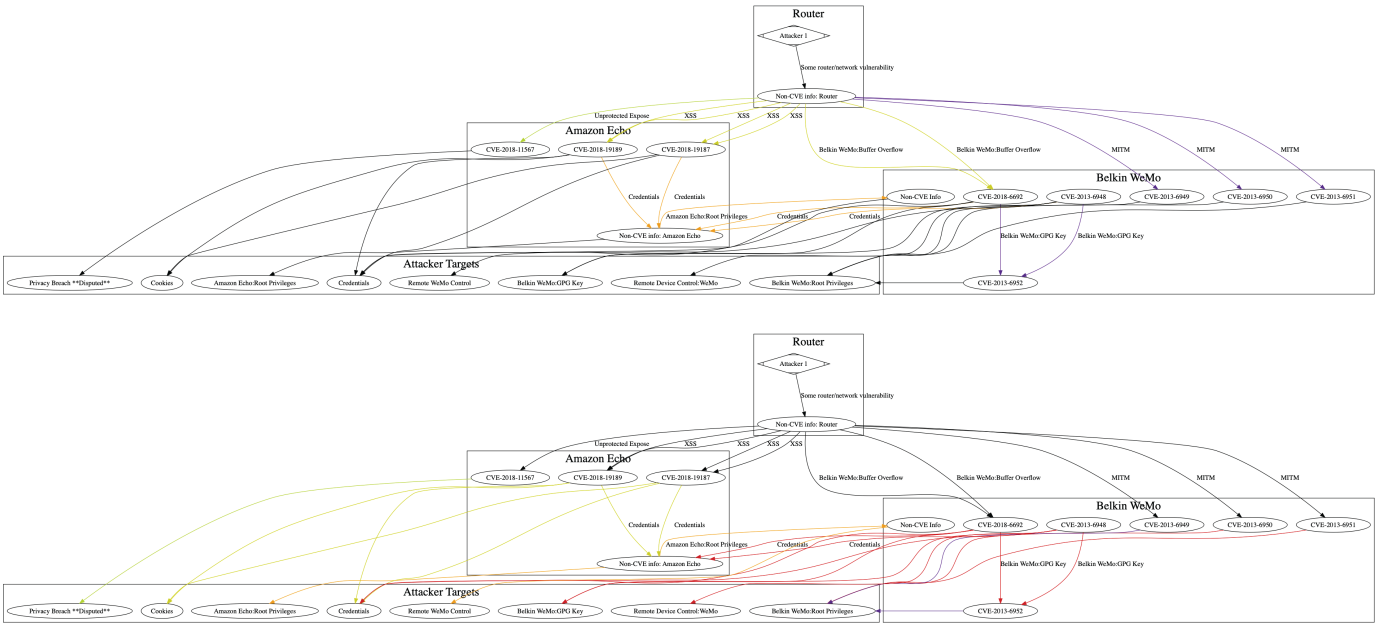

Fig. 7. Exploitability (top) and impact (bottom) circuits corresponding to two devices under the observation of all vulnerabilities. Full-sized images are available in the repository.

(an Amazon Echo Dot) with one vulnerability (CVE-2018-11567), one consisting of the Amazon Echo Dot and all of its corresponding CVEs, and one consisting of the Amazon Echo Dot, a Belkin WeMo smart plug, and all of their associated CVEs. The generated exploitability and impact circuits for each of the three settings are shown at the back of this paper in Figures 5, 6, and 7, respectively. Note that the attack circuit complexity grows exponentially with respect to the number of devices in the network, which is visually evident in these figures and is the reason why we primarily focus on smaller networks to demonstrate our work.

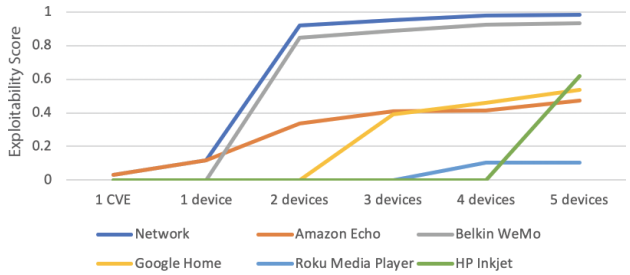

Fig. 8. Exploitability score adjustment with the addition of new devices (augmented in sequence (left to right, top to bottom)) and CVEs.

In Table I, we record the scores of devices and the network as they were calculated with each of the networks. These results reflect the dynamic activity metrics that we recorded in our experimental observation of the devices over a period of 4 days, and the output values respond accordingly as we experimentally change the dynamic metrics of the devices (for instance, in contrast to the 1-CVE, 1-device network yielding an exploitability score of 0.0289 with the Echo's actual status of "`rarely_online`", we noted that the exploitability score rose to 0.0378 when the device was marked as "`frequently_online`"). Note that where only one device is used, the Belkin WeMo scores are N/A because the device is not present in that network. We select these particular devices to evaluate and illustrate the change in scores when two devices of different levels of vulnerability are added to a network. Based on its three total CVEs, the Amazon Echo's vulnerabilities don't yield as high of a risk, exploitability, or impact score of the device. Its CVEs CVE-2018-11567, CVE-2018-19189, and CVE-2018-19187 have base exploitability scores of 1.8, 2.8, and 2.8, and base impact scores of 1.4, 2.7, and 2.7, respectively. In contrast, three of the Belkin WeMo's six CVEs (CVE-2018-6692, CVE-2013-6952, and CVE-2013-6949) are serious vulnerabilities with base exploitability scores of 3.9, 10.0, and 8.6, and base impact scores of 6.0, 10.0, and 10.0, respectively. Note that in the first column, we observe a network with a vulnerability that is of relatively lesser concern (CVE-2018-11567, with a base exploitability score of 1.8 and base impact score of 1.4). When more vulnerabilities are added in, the device's scores all increase as expected, and the network score increases in the same way (because in this setting, the network is defined solely by the device). When the WeMo is added, several paths between the WeMo's vulnerabilities and the Echo's vulnerabilities are discovered from the I/O mapping step, which causes each of the Echo's scores to rise, and the network's scores likewise undergo an increase with the introduction of these new vulnerabilities.

This then raises the following question: how can the scores of networks, which are largely heterogeneous and unique, be interpreted relativistically? This is a difficult problem to

	Echo, 1 CVE	Echo, all CVEs	Echo, WeMo
E_{Echo}	0.0289	0.1182	0.3380
I_{Echo}	0.0140	0.0679	0.1776
Echo R_{Conf}	0.0073	0.0341	0.0982
Echo R_{Integ}	0.0	0.0268	0.0910
Echo R_{Avail}	0.0	0.0	0.0644
E_{WeMo}	N/A	N/A	0.8490
I_{WeMo}	N/A	N/A	0.4823
WeMo R_{Conf}	N/A	N/A	0.5744
WeMo R_{Integ}	N/A	N/A	0.5649
WeMo R_{Avail}	N/A	N/A	0.4605
$E_{Network}$	0.0289	0.1182	0.9223
$I_{Network}$	0.0140	0.0679	0.6078
Network R_{Conf}	0.0073	0.0341	0.6367
Network R_{Integ}	0.0	0.0268	0.6239
Network R_{Avail}	0.0	0.0	0.5098

TABLE I
DEVICE AND NETWORK SCORES FOR DIFFERENT NETWORK SETTINGS.

address, especially without data from real attack circuit use. For now, we normalize the score to a range of $[0, 1]$ using a sigmoidal ($\tanh$) function. In general, as the number of devices grows and the vulnerabilities increase, the score trends will demonstrate a sigmoidal behavior, converging to 1. In Figure 8, we observe one metric—the exploitability score—of 5 devices and the overall network, and how it changes as more devices are added. In particular, we start with the Amazon Echo with one vulnerability, then add the rest of the Amazon Echo's vulnerabilities, then add the Belkin WeMo, the Google Home Mini, the Roku media player, and finally the HP Inkjet printer in sequence. As expected, the network's exploitability score approaches a value of 1 as more devices are added. The case may be made that this scoring method is too sensitive (regardless of the fact that these networks are comprised of devices whose vulnerabilities we know). This may be a result of the dampener and normalization variable values or lack of information about attack circuit behavior in practice. We leave the related refinement of the scoring method to future work.

VII. CONCLUSION

In this paper, we address the problem of evaluating the security of a network. This is done by using *attack circuits* and associated compositional scores and dynamic activity metrics. In this manner, an individual IoT device or network may be analyzed for its vulnerability to security attacks. Evaluation in Section VI demonstrates the increased security risks for a growing IoT network. While our work focuses on using descriptions to extract *input/output* pairs, this approach may be extended to extract multiple pairs per description, as well as using other available information sources. Activity metrics may be further developed by using generative machine learning models to learn abnormal network traffic behaviors. We also noted that as networks grow, their complexity grows exponentially. Thus, the network flow problems described may become inefficient with huge IoT attack circuits, and alternative approaches like graph neural networks (GNNs) [30] may be required for their analysis. However, for application in smaller IoT networks (e.g. smart homes), we conclude that this approach suffices.

REFERENCES

[1] "Smart home," https://www.statista.com/outlook/279/109/smart-home/united-states, 2019.

[2] "Internet of things (iot) connected devices installed base worldwide from 2015 to 2025 (in billions)," https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/, 2019.

[3] P. Mell, K. Scarfone, and S. Romanosky, "Common vulnerability scoring system," *IEEE Security & Privacy*, vol. 4, no. 6, 2006.

[4] R. A. Martin, "Common weakness enumeration," *Mitre Corporation*, 2007.

[5] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and A. S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" *arXiv preprint arXiv:1808.02741*, 2018.

[6] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic," *arXiv preprint arXiv:1708.05044*, 2017.

[7] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," *arXiv preprint arXiv:1705.06805*, 2017.

[8] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2017, pp. 2177–2184.

[9] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE, 2009, pp. 117–126.

[10] D. Turner, S. Entwisle, O. Friedrichs, D. Ahmad, D. Hanson, M. Fossi, S. Gordon, P. Szor, E. Chien, D. Cowings *et al.*, "Symantec internet security threat report: trends for july 2004-december 2004," *Retrieved July*, vol. 30, p. 2005, 2005.

[11] K.-K. R. Choo, "The cyber threat landscape: Challenges and future research directions," *Computers & Security*, vol. 30, no. 8, pp. 719–731, 2011.

[12] F. R. Chang, "Is your computer secure?" *Science*, vol. 325, no. 5940, pp. 550–551, 2009.

[13] S. Noel and S. Jajodia, "Optimal ids sensor placement and alert prioritization using attack graphs," *Journal of Network and Systems Management*, vol. 16, no. 3, pp. 259–275, 2008.

[14] R. P. Lippmann and K. W. Ingols, "An annotated review of past papers on attack graphs," MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, Tech. Rep., 2005.

[15] Y. Matsuoka, D. Sloo, M. Veron, S. Honjo, I. Guenette, and M. R. Malhotra, "Security scoring in a smart-sensored home," Mar. 5 2015, uS Patent App. 14/489,162.

[16] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of massive datasets*. Cambridge university press, 2014.

[17] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[18] P. Nathan, "Pytextrank, a python implementation of textrank for text document nlp parsing and summarization," https://github.com/ceteri/pytextrank/, 2016.

[19] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[21] A. Goldberg, E. Tardos, and R. Tarjan, "Network flow algorithms," dept. of computer science," Stanford Univ., Technical Report STAN-CS-89-1252, Tech. Rep., 1989.

[22] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[23] K. S. Jones, *Readings in information retrieval*. Morgan Kaufmann, 1997.

[24] L. R. Ford and D. R. Fulkerson, "Maximal Flow through a Network." *Canadian Journal of Mathematics*, vol. 8, pp. 399–404. [Online]. Available: http://www.rand.org/pubs/papers/P605/

[25] G. Dantzig and D. R. Fulkerson, "On the max flow min cut theorem of networks," *Linear inequalities and related systems*, vol. 38, pp. 225–231, 2003.

[26] S. Bird and E. Loper, "Nltk: the natural language toolkit," in *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*. Association for Computational Linguistics, 2004, p. 31.

[27] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[28] J. Leskovec and R. Sosič, "Snap: A general-purpose network analysis and graph-mining library," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 8, no. 1, p. 1, 2016.

[29] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, 2006.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, 2009.