Helion: Enabling Natural Testing of Smart Homes

Prianka Mandal
William & Mary
Williamsburg, Virginia, USA
pmandal@wm.edu

Sunil Manandhar
IBM Research
Yorktown Heights, NY, USA
sunil@ibm.com

Kaushal Kafle
William & Mary
Williamsburg, Virginia, USA
kkafle@wm.edu

Kevin Moran
University of Central Florida
Orlando, Florida, USA
kpmoran@ucf.edu

Denys Poshyvanyk
William & Mary
Williamsburg, Virginia, USA
denys@cs.wm.edu

Adwait Nadkarni
William & Mary
Williamsburg, Virginia, USA
apnadkarni@wm.edu

ABSTRACT

Prior work has developed numerous systems that test the security and safety of smart homes. For these systems to be applicable in practice, it is necessary to test them with realistic scenarios that represent the use of the smart home, *i.e.,* home automation, in the wild. This demo paper presents the technical details and usage of Hɛlion, a system that uses n-gram language modeling to learn the regularities in user-driven programs, *i.e.,* routines developed for the smart home, and predicts *natural* scenarios of home automation, *i.e.,* event sequences that reflect realistic home automation usage. We demonstrate the HɛlionHA platform, developed by integrating Hɛlion with the popular Home Assistant platform. HɛlionHA allows an end-to-end exploration of Hɛlion's scenarios by executing them as test cases with real and virtual smart home devices.
The demo video can be found here: https://youtu.be/o9g0wKiJJMI

CCS CONCEPTS

• **Security and privacy**;

KEYWORDS

Home Automation, Trigger-Action Programming, Language Models, Home Assistant

ACM Reference Format:
Prianka Mandal, Sunil Manandhar, Kaushal Kafle, Kevin Moran, Denys Poshyvanyk, and Adwait Nadkarni. 2023. Helion: Enabling Natural Testing of Smart Homes. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '23), December 3–9, 2023, San Francisco, CA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3611643.3613095

1 INTRODUCTION

In smart home platforms, automation is driven by trigger-action programs known as *routines*, wherein a certain *action* event is programmed to occur after a certain *trigger*, *e.g.,* IF the user is home (trigger), THEN turn the camera off (action). Prior work has analyzed routines created by developers (*i.e.,* IoT apps) to understand the security and safety issues in home automation [2–4, 8–10, 17]. However, IoT apps defined by developers may not reflect realistic home automation use in the wild, *i.e.,* the events that are likely to actually occur in end-user homes. The unavailability of realistic home automation usage makes it difficult to design or evaluate systems designed to analyze/test home automation in a practical manner.

For instance, consider the problem of testing the effectiveness of a security analysis/system for evaluating home automation. At present, researchers generally evaluate their systems with random permutations of smart home events [4, 8, 17], which may not represent realistic home automation usage in the wild, and lead to an impractical design or evaluation of the systems. One possible solution to this problem would be to collect real execution traces of smart home events from end-user homes, and then use those traces to build/evaluate systems. However, not only is this approach extremely privacy-invasive (*i.e.,* as the traces also represent physical events in the user's home), but may also be ineffective, since the traces may contain superfluous events that represent platform and device-specific intricacies, *i.e.,* "noise", which may distract from the real, semantically-relevant, smart home usage. Thus, there is a need for synthetically generated but realistic home automation scenarios that can be used to generate effective test cases.

We previously built a framework, Hɛlion [12], that leverages *user-driven routines*, *i.e.,* routines created by end-users using simple trigger-action user interfaces provided by most popular smart home platforms (*e.g.,* NEST [13], SmartThings [15]). User-driven routines represent the real home automation requirements of users, as they allow end-users to express their home automation workflows/programs via the UI, without writing a single line of code, and hence, eliminating the need for (or relevance of) developer-provided IoT apps. That is, routines obtained from a user, combined with simple cues regarding their order/frequency of execution, form the "home automation program" for that user. Hɛlion builds upon prior work in the SE domain on leveraging the naturalness in code for tasks such as code completion [6], and similarly, learns the regularities in a corpus of such home automation programs derived from end-users using n-gram LMs. Hɛlion's model can then be used as a sequence generator to predict natural scenarios, *i.e.,* realistic home automation event sequences based on a given history of events. These natural scenarios can then be used as test cases in the design and evaluation of security systems built for the smart home, *e.g.,* in lieu of random events used by prior work. The full details of Hɛlion's methodology, design considerations, evaluation results,

ESEC/FSE '23, December 3–9, 2023, San Francisco, CA, USA

ACM ISBN 979-8-4007-0327-0/23/12.
https://doi.org/10.1145/3611643.3613095

and discussion of the findings are described in our past study [12]. The source code of Hɛlion is also publicly available on GitHub [16].

This demo paper describes the implementation and usage of Hɛlion, and particularly, the implementation of HɛlionHA, an extension to the popular Home Assistant platform [7] with Hɛlion, which enables users to generate natural scenarios as well as automatically execute them as test cases with real/virtual devices connected to the platform. HɛlionHA first provides a UI for configuring Hɛlion's model and automatically generating scenarios adhering to the configuration. These scenarios are in the form of sequences of event tokens predicted by Hɛlion. HɛlionHA converts the event tokens from the sequences into Home Assistant-specific events, and passes the events on to the system by interfacing with relevant components (specifically, the Event Bus), and in this manner, *executes* the scenario on the platform. HɛlionHA can be connected to physical devices, or virtual devices configured via the UI, to execute a wide variety of Hɛlion's scenarios. HɛlionHA's dashboard also enables the user to monitor various smart home states during the execution of a scenario. The source code and documentation of Hɛlion on Home Assistant, *i.e.,* HɛlionHA, are publicly available [1].

The paper is organized as follows: Section 2 discusses the necessary background on language modeling and the fundamentals of Home Assistant. Section 3 provides a brief overview of Hɛlion. Section 4 presents the detailed design and implementation of Hɛlion and Section 5 presents the integration of Hɛlion into Home Assistant, *i.e.,* HɛlionHA. Finally, Section 6 concludes the paper.

2 BACKGROUND

This section provides the rationale behind choosing the n-gram language model for Hɛlion and a brief overview of Home Assistant.

2.1 N-gram Language Model

Hindle *et al.* demonstrated that source code written by humans is just like natural language and thus, contains patterns that make it predictable [6]. Similarly, user-driven routines are also natural as these are created by humans to fulfill a particular workflow, and hence, may contain regularities/patterns that are predictable. Based on this, we use statistical language modeling to learn the regularities in home automation sequences composed of user-driven routines (*i.e.,* which represent the end-user's overall "program"), and use the model to predict natural scenarios of home automation.

In general, language models (LMs) measure the probability of a sentence $s = w_1^m = w_1 w_2 ... w_m$, given the probabilities of the individual words in the sentence (*i.e.,* w_1^m), as previously estimated from a training corpus. This ability enables prediction, *i.e.,* predicting the next most probable word that can follow a sequence of words, known as the "context" or "history". When modeling smart home routines, we define a "sentence" to represent a sequence of home automation events, wherein the "words" (a.k.a *tokens*) are smart home events (*e.g.,*<LightBulb, switch, ON>).

In Hɛlion, we specifically use n-gram LMs as they assume the Markov property, *i.e.,* instead of computing the conditional probability given an entire event or language history, we can approximate it by considering only a few tokens from the past as the history. The intuition behind n-gram LMs applied to natural language is that shorter sequences of words are more likely to co-occur in training corpora, thus providing the model with more examples to condition token probabilities, enhancing its predictive power. Using the n-gram model, we estimate the probability of the event sequence $s = e_1^m = e_1 e_2 ... e_m$ as follows:

$$p(e_1^m) = \prod_{i=1}^{m} p(e_i | e_1^{i-1}) \approx \prod_{i=1}^{m} p(e_i | e_{i-n+1}^{i-1}) \quad (1)$$

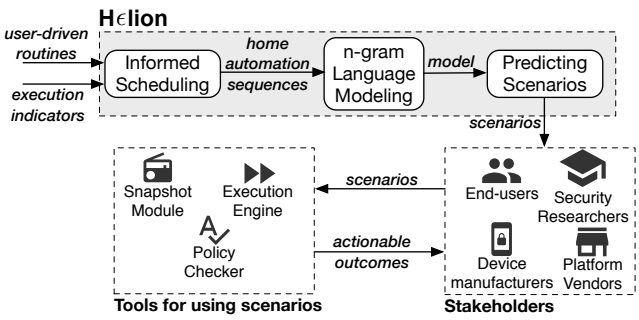

Figure 1: An overview of the Hɛlion framework, which models home automation sequences to construct natural scenarios. Stakeholders use tools that analyze or execute scenarios to obtain actionable outcomes.

2.2 Home Assistant

Home Assistant is an open-source software platform for home control and automation [7]. Here we briefly describe the key areas of Home Assistant that are necessary for integrating Hɛlion into it to develop HɛlionHA.

The Home Assistant **dashboard:** The dashboard is a customizable page where users can manage their home using HomeAssistant's mobile and Web interfaces. The overview dashboard is the first thing that users see after installing Home Assistant. The dashboard displays information connected to and available in Home Assistant, including the connected devices, both real and virtual.

Cards: The Home Assistant Dashboard is composed of cards. Each card has its own configuration options, which users can configure as required. Moreover, users can build and use their own cards. One of the most common cards is the "entities" card which groups abstract items together into lists.

Configuration: Other than the user interface, users can configure their Home Assistant instance by editing configuration.yaml. The configuration.yaml file contains integrations to be loaded along with their configurations.

3 HELION

Figure 1 shows Hɛlion, a data-driven framework that models the regularities of user-driven home automation, generates natural home automation scenarios, and provides stakeholders with tools to use the scenarios and obtain actionable outcomes.

The first step of Hɛlion is **data collection**, *i.e.,* collecting *user-driven routines* from users along with *execution indicators*. Execution indicators are clues about when or how frequently those user-driven routines are scheduled to execute. The next step is **informed scheduling** where the routines and execution indicators

are transformed into home automation *event sequences*, which represent the user's "program" for a certain duration (*e.g.,* a month). During the **modeling** phase, Hɛlion uses n-gram LMs on the corpus of event sequences obtained from users. Finally, during **scenario prediction**, Hɛlion generates natural scenarios, which can be used by stakeholders for testing the smart home, such as executable *test cases* in HɛlionHA to be executed with real and virtual devices.

4 DESIGN AND IMPLEMENTATION

This section describes the design and implementation of Hɛlion, elaborating on its four steps: data collection and representation, informed scheduling, modeling, and scenario prediction.

4.1 Data Collection and Representation

The most practical way to collect user-driven routines is collecting routines directly from users. To do this, we conducted a user survey with 40 users and obtained 273 routines (233 unique) created by the users. After collecting data from users, we transformed those into home automation tokens. Anonymized datasets and code for the Hɛlion are available at [16].

Collecting Data from Users: We used the survey to collect the routines from users. In the survey, participants were asked to create routines as in the "IF" and "THEN" trigger-action format, but expressed in plain English text, allowing users to express any functionality desired without enforcing artificial constraints. Here is the raw routine from our dataset:

IF the motion is detected **THEN** camera takes a picture

After creating routines, participants specified execution indicators, *i.e.,* the time-range, day-range, and frequency indicators for their routines (described in detail in Section 4.2).

Representing smart home events as tokens: A token is a home automation event parsed from a structured natural language routine. Here, an event can denote a change in the state of a device (*e.g.,* door locked) or the home (*e.g.,* the user is away). We define Hɛlion's home automation event token as:

$$e_i :=< device_i, attribute_i, action_i >$$

where $device_i$ represents the device (*e.g.,* door lock, camera), the $attribute_i$ corresponds to one of a predefined set of device attributes (*e.g.,* the *lock* attribute for the door lock, which can take the values Locked/Unlocked), and $action_i$ represents the change of state, and hence, the current value of $attribute_i$. Using this design, the example routine discussed previously (*i.e.,* the motion sensor/camera) would be tokenized as (terms from SmartThings capabilities [14]):

$$< motion\_sensor,\ motion, \text{DETECTED}>,$$
$$< security\_camera,\ image, \text{TAKE}>$$

4.2 Informed Scheduling

A home automation event sequence is an approximate ordered representation of how the routines would execute in the user's home in a particular timeframe. Hɛlion transforms the tokenized routines specified by a particular user into a home automation event sequence. Here, the order is important for generating home automation sequences. Therefore, we introduce a novel abstraction for users to stipulate the approximate order in which routines may execute, *i.e.,* routine-specific execution indicators. Users may be able to describe when they perform certain personal tasks which trigger home automation,*i.e.,* when they come home, go to work, bed, cook, or do laundry. Execution indicators allow us to capture such factors, which we then leverage to schedule routines to create home automation sequences. This is why we define the approach as informed scheduling, as the scheduling mechanism is informed by the user's understanding of their own home use.

Execution indicators constitute the time and frequency of the potential execution of a routine. For Hɛlion, we consider three types of indicator: (1) the time-range indicator (*e.g.,* early morning, noon, and night), (2) the day-range indicator (*e.g.,* mostly on weekdays, and mostly on weekends), and (3) the frequency indicator (*e.g.,* many times a day, few times a day, few times a month). Routines collected from users have been scheduled in the time series based on these indicators, also provided by the users, we extract the ordered set of routines from this month-long series as the execution sequence and construct the HOME corpus. The HOME corpus consists of 30,518 events, from 40 month-long sequences (*i.e.,* 40 users), generated from 273 routines (233 unique) and their execution indicators.

4.3 Modeling

Hɛlion uses the n-gram LM to learn the regularities in user-driven home automation sequences. For the Hɛlion's n-gram model, we choose $n \geq 3$. Choosing values of $n < 3$ can either completely ignore the context or capture simple relationships that are already observable from data. Considering larger values of n, *i.e.,* $n \geq 3$, the model can learn non-obvious regularities in home automation corpora. However, considering too much of the event history (*i.e.,* a very large n) may actually hurt the predictive power of the model. Moreover, longer sequences may be relatively uncommon in the wild, even if they are realistic and useful for uncovering serious security/safety flaws. Therefore, choosing $n \geq 3$ leads to a better model. We used interpolated smoothing methods since it performs well with lower-order (*i.e.,* 3-4 gram) models [5].

4.4 Scenario Prediction

Hɛlion considers the language model as a sequence generator that can produce an arbitrarily long series of events, *i.e.,* home automation scenarios.

For security and safety-related testing, researchers require both natural scenarios that are reasonably likely to occur in end-user homes and unnatural scenarios that demonstrate unsafe situations. Hence, we designed Hɛlion to generate two corresponding flavors of scenarios:

The *up* flavor, natural scenarios: This is the default flavor where Hɛlion predicts the most probably event(s) given a history, *i.e.,* generates natural home automation scenarios.

The *down* flavor, unrealistic scenarios: The down flavor generates unrealistic/unnatural scenarios by predicting the most improbable event(s) given a particular history.

5 IMPLEMENTATION OF HELIONHA

We implemented HɛlionHA by integrating Hɛlion with Home Assistant. In this section, we describe how we implement Hɛlion on the Home Assistant platform to develop HɛlionHA as well as how users can use it.

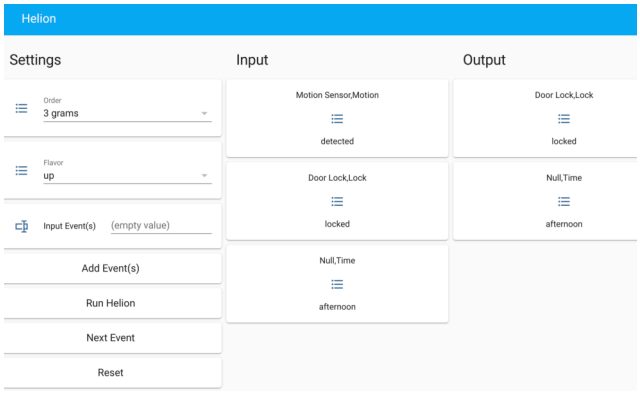

Figure 2: An overview of Hɛlion on Home Assistant platform.

5.1 Conversion of Hɛlion Tokens to Entities and Cards in Home Assistant

Entities in Home Assistant are abstract objects that hold the state of the simulated device, *i.e.,* each entity is Home Assistant's representation of the function of a device. To elaborate, Home Assistant allows users to connect to physical devices, and usually, entities only serve as the interface to those devices. As we defined in Section 4, a token is a Hɛlion's representation of a device, its attribute, and its state (*i.e.,* token, $e_i = < device_i, attribute_i, action_i >$). To implement HɛlionHA, we needed to convert each token to an entity name. We implemented the `parse_token.py` script which takes in a token as an input and returns the corresponding entity. If the token contains multiple devices, then each entity name will be separated by a space. This entity is then added to the `ui-lovelace.yaml` or `helion.yaml` file.

We used two types of entities in HɛlionHA: (i) `input_boolean` for keeping track of two states (*e.g.,* lightbulb has two states: on or off). (ii) `input_select` for keeping track of multiple or complex states (*e.g.,* the motion sensor has four states: activated, deactivated, detected, and not_detected).

Cards are the components that are displayed in the Home Assistant dashboard and represent entities. The state of the entity can be seen or changed through cards. In our implementation, each card is set up to correspond to an entity. Users can also input a token through an input card which will run a script to modify `helion.yaml`. We implemented a script `change_ui_cards.py`, that is used to create the cards that are displayed on the Home Assistant dashboard. When tokens are output from the Hɛlion server, we take the tokens that are output and pass them to this script, which finds the corresponding card and modifies `helion.yaml`.

5.2 Predicting Scenarios using the HɛlionHA UI and Executing them as Test Cases

In this section, we describe how exactly Hɛlion integrates with Home Assistant, in terms of both the user interface (*i.e.,* the HɛlionHA dashboard) as well as the backend (*i.e.,* the Hɛlion server).

We created a custom page on Home Assistant where the UI specific to Hɛlion resides. The new page is built on top of Lovelace UI which is a customizable Home Assistant dashboard [11]. Figure 2 shows the UI of HɛlionHA. The UI of HɛlionHA has three main parts: Settings, Input, and Output, each consisting of several cards.

The user first specifies the settings, particularly the order (3-gram or 4-gram), and flavor (up or down). These elements are stored as `input_select` entities in the backend. The user then specifies the event history, *i.e.,* the input events that form the context following which Hɛlion predicts future events. When "Run Helion" is clicked on the dashboard UI, HɛlionHA takes the user-provided history/input events from the `input_list` and reprocesses them to send to the Hɛlion server (which executes in the backend). The Hɛlion server then provides predictions given the settings and inputs, and HɛlionHA transforms them into Home Assistant events, and displays the same in the dashboard.

To elaborate, the scenario, *i.e.,* predictions generated by the Hɛlion server are gathered in up.tsv and down.tsv based on which flavor the user requested. Note that these predicted events are Hɛlion's tokens, *i.e.,* in the form $< device, attribute, state >$, which must now be converted to HɛlionHA's events, to be executed with real and virtual devices on the platform.

To generate each Home Assistant event (represented as [device.attribute, state]) corresponding to a predicted token, we invoke HɛlionHA's `call_service` method, which takes the entity_id (*i.e.,* the device) and the target_state (*i.e.,* the state) as parameters, along with the method to call (*i.e.,* the attribute) to properly set the entity_id to its target_state. This is equivalent to sending a command to a physical (or virtual) device. In essence, after the call_service method is invoked, the device.attribute in the Home Assistant token is set to the state specified within that token. When the token is executed, it is broadcast as an event in Home Assistant's Event Bus to let other entities (and automation) know that a state change has occurred. Finally, it reloads the UI to display the updated states of the entities.

6 CONCLUSION

This tool demonstration paper describes the implementation and usage of Hɛlion, a framework for predicting natural scenarios for home automation to enable the testing of security and safety solutions built for the smart home. Further, we develop and describe Hɛlion's extension to the Home Assistant platform, HɛlionHA, which allows end-users to generate diverse scenarios with various model parameters (*e.g.,* varying the n or the prediction flavor), and automatically execute the scenarios as test cases with real and virtual devices.

ACKNOWLEDGMENTS

The authors have been supported in part by the NSF-CNS-2132281 grant. The authors acknowledge the contributions from Amit Seal Ami for his help in finalizing the artifact. Moreover, the following undergraduate students from William & Mary contributed in developing HɛlionHA: Hannah Cummings, Kaitlin Haynal, Kevin Jiao, Jessica Pesso, Kyoko Minamino, Sarah Wang, and Connor Yu.

REFERENCES

[1] Hɛlion. 2023. Implementation of Hɛlion on Home Assistant. https://github.com/Secure-Platforms-Lab-W-M/Helion-on-Home-Assistant.
[2] Z Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A Selcuk Uluagac. 2018. Sensitive Information Tracking in

Commodity IoT. In *Proceedings of the 27th USENIX Security Symposium (USENIX)*.
[3] Z. Berkay Celik, Patrick McDaniel, and Gang Tan. 2018. Soteria: Automated IoT Safety and Security Analysis. In *2018 USENIX Annual Technical Conference (USENIX ATC)*. 147–158.
[4] Z. Berkay Celik, Gang Tan, and Patrick McDaniel. 2019. IoTGuard: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *2019 NDSS Symposium. To appear.*.
[5] Stanley F Chen and Joshua Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language* 13, 4 (1999).
[6] A. Hindle, E.T. Barr, Z. Su, M. Gabel, and P. Devanbu. 2012. On the Naturalness of Software. In *International Conference on Software Engineering (ICSE'12)*. 837–847.
[7] Home Assistant. Accessed May 2023. Home Assistant. https://www.home-assistant.io/.
[8] Yunhan Jack Jia, Qi Alfred Chen, Shiqi Wang, Amir Rahmati, Earlence Fernandes, Z Morley Mao, Atul Prakash, and Shanghai JiaoTong Unviersity. 2017. ContexIoT: Towards providing contextual integrity to appified IoT platforms. In *Proceedings of the 2017 Network and Distributed System Security Symposium (NDSS)*.
[9] Kaushal Kafle, Kevin Moran, Sunil Manandhar, Adwait Nadkarni, and Denys Poshyvanyk. 2019. A Study of Data Store-based Home Automation. In *Proceedings of the 9th ACM Conference on Data and Application Security and Privacy (CODASPY)*.
[10] Kaushal Kafle, Kevin Moran, Sunil Manandhar, Adwait Nadkarni, and Denys Poshyvanyk. 2020. Security in Centralized Data Store-based Home Automation Platforms: A Systematic Analysis of Nest and Hue. *ACM Transactions on Cyber-Physical Systems (TCPS)* 5, 1 (Dec. 2020).
[11] Lovelace UI. 2023. Lovelace UI released! https://www.home-assistant.io/blog/2019/01/23/lovelace-released/.
[12] Sunil Manandhar, Kevin Moran, Kaushal Kafle, Ruhao Tang, Denys Poshyvanyk, and Adwait Nadkarni. 2020. Towards a natural perspective of smart homes for practical security and safety analyses. In *2020 ieee symposium on security and privacy (sp)*. IEEE, 482–499.
[13] Nest Labs. Accessed June 2018. Works with Nest. https://nest.com/works-with-nest//.
[14] SmartThings Developer Documentation. Accessed December 2018. Capabilities Reference. https://docs.smartthings.com/en/latest/capabilities-reference.html.
[15] Smartthings Developers. Accessed June 2018. Documentation. http://developer.smartthings.com/.
[16] Sunil Manandhar. 2020. Helion Data and Source Code. https://github.com/helion-security/helion.
[17] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. 2018. Fear and Logging in the Internet of Things. In *Network and Distributed Systems Symposium*.

Received 2023-05-11; accepted 2023-07-20