A Flexible and Generic Functional Mock-up Unit Based Threat Injection Framework for Grid-interactive Efficient Buildings: A Case Study in Modelica

Yangyang Fu[a], Zheng O'Neill[a,], Veronica Adetola[b]

[a]*J. Mike Walker'66 Department of Mechanical Engineering, Texas A&M University, College Station, TX, USA*
[b]*Pacific Northwest National Laboratory, Richland, WA, USA*

Abstract

Grid-interactive efficient buildings (GEBs) have been considered as an important asset to support the power grid reliability by utilizing the demand flexibility offered by GEBs. GEBs are enabled by advances in sensors and controls, and the communication between building equipment, whole buildings, and the grid. The integration of different building technologies and network-based communication system makes GEBs vulnerable to passive threats such as equipment failure and active threats such as cyber-attacks. Modeling and simulation is an effective way to evaluate the impact of threats on the system performance. This paper proposes a generic and flexible threat injection framework for commonly-used building energy simulators such as EnergyPlus and Modelica to support threat modeling and evaluation. This framework leverages functional mock-up unit (FMU) to develop a general modeling interface for threat injection and simulation. A numerical case study using Modelica as a building energy simulator is conducted to demonstrate the capability of the framework for supporting single/multiple-order threat modeling and simulation of a GEB. Four threats and their combinations are injected on a Modelica-based threat-free building energy and control system, including operating supply fan at its full speed, remotely cycling the chiller on and off, blocking the chiller from receiving the chilled water supply temperature setpoints, and hijacking the global zone air temperature setpoint. Simulation results show that the cyber-attack that leads to short-term signal blocking has small effects on the system operation due to the "self-healing" feature of the heating, ventilation, and air-conditioning (HVAC) interactive control system. The threat that takes control of resetting the global zone air temperature setpoints has the most adverse impact on the system energy use, peak power demand, thermal comfort and the provision of demand flexibility. The combination of four threats have aggregative effects on the system but the effects are less than the additive effects of the individual threat.

Keywords: Threat Injection, Functional Mock-up Unit, Cyber-attack, Building Control, Grid-interactive Efficient Buildings

Nomenclature

Abbreviations

AHU	Air handling unit
ASHRAE	American society of heating, refrigerating and air-conditioning engineers
BACnet	Building automation and control networks
BAS	Building automation system
CHWST	Chilled water supply temperature
DFI	Demand flexibility index
DoS	Denial-of-service
EMS	Energy management system
FDD	Fault detection and diagnosis
FMI	Function mock-up interface
FMU	Functional mock-up unit
GEB	Grid-interactive efficient building
HVAC	Heating, ventilation and air conditioning

Email addresses: yangyang.fu@tamu.edu (Yangyang Fu), zoneill@tamu.edu (Zheng O'Neill), veronica.adetola@pnnl.gov (Veronica Adetola)

KPI	Key performance index
PID	Proportional-integral-derivative
SAT	Supply air temperature
VAV	Variable air volume

Subscripts

e	End time
max	Maximum value
min	Minimum value
ref	Reference value
s	Start time
z	Zone index in a multi-zone building

Symbols

$\mathbb{A}$	Threat period
d	Temperature deviations from the lower and upper bounds
D	Total discomfort
$\overline{DFI}$	Upward demand flexibility index
$\underline{DFI}$	Downward demand flexibility index
E	Energy usage
$f_p(.)$	Pulse function
k	Constant
P	Instant system power
$\overline{P}$	Upward instant power trajectory
$\underline{P}$	Downward instant power trajectory
y	True value
$\hat{y}$	Attacked value
$\mathbb{Z}$	Multi-zone building
Δt	Time step

1. Introduction

The electrical grid is facing multiple challenges: increasing peak electricity demand, stability and resilence concerns due to high penetration of variable renewable electricity generation, and transmission and distribution infrastructure constraints. These challenges stress the electrical grid by making it difficult to balance the supply and the demand for different time scales under the constraints of current grid infrastructure. Balance management services are largely provided by supply-side entities: integrated utilities, grid operators, and generators. Demand-side entities such as buildings with flexible electrical loads can also be utilized for power balancing, but unfortunately, they have yet to be fully investigated. In 2017, the buildings sector consumed about 76% of electricity and was responsible for 40% of all U. S. primary energy use and associated greenhouse gas emissions [1]. In 2019, the Department of Energy in the United States released a series of reports about grid-interactive efficient buildings (GEBs), and how GEBs can be used to provide grid services [2]. The motivation behind these reports is that buildings can proactively and intelligently provide the demand flexibility using different assets in response to price changes or direct grid signals.

Sensors and controls, connected devices, and associated communication network play vital roles in GEBs, which creates an enormous opportunity for threats. GEBs suffer from passive threats such as physical faults and active threats such as cyber-attacks maliciously launched on the system. A passive threat refers to one that is not injected intentionally by human beings such as outdoor air damper stuck in the heating, ventilation, and air condition (HVAC) system. Malfunctioning building equipment, sensors and control systems, are considered as the top causes for "deficient" building systems. These malfunctions cause significant energy waste [3] and occupant discomfort [4]. An active threat refers to any incident that is deliberately created with human actively engaged such as cyber-attacks through the network by hackers. The building automation system (BAS), serving as the brain for GEB operation, includes a cyber-infrastructure of sensing, computation, communication, and control components for the close monitoring and control of the mechanical components and physical environment. The vulnerability of GEBs under cyber-attacks comes from many aspects. First, as the most popular communication protocol for the BAS in commercial buildings, Building Automation and Control networks (BACnet) protocol was not designed with security as a primary requirement, as the original intention and implementation of BAS was isolated from the external connection. With the advancement of networking technology, BAS networks now are connecting to internal enterprise networks for remote management and cloud-based data analytics. As such, the attack surface against BAS networks in buildings, including BACnet

managed networks has increased [5]. In 2013, researchers discovered security vulnerabilities in Tridium Niagara, a widely-used commercial BAS, and successfully hacked the Tridium system in Google's Sydney headquarter [6]. A massive credential data theft at Target in 2014 began with attackers compromising the HVAC systems [7]. Second, as more devices and software systems interconnected and interacted, vulnerability in one component can be used to access the data, attack, and/or compromise other components. If buildings and the grid are more tightly integrated and connected as in GEBs, vulnerabilities in building software and devices could be used to attack the larger grid. Even if the grid is not directly compromised, the grid that is more heavily reliant on building-based services to maintain the stability is indirectly made more vulnerable by greater building-level automation and connectivity [8].

Modeling and simulation is an effective way to evaluate the impact of threats on the system. The evaluation can be used to properly design threat diagnosis or defending algorithms. This paper is focusing on a threat injection framework for the evaluation and impact analysis on GEBs. A detailed literature review towards threat modeling and evaluation in the building energy system and related areas is provided in Section 1.1 and the current research gaps in terms of modeling and evaluating threats in the building energy and control system are summarized in Section 1.2.

1.1. Literature Review

Threat modeling is an effective way to generate bulk data containing a variety of threats with the ground truth under different operating conditions. It can also generate data for concurrent threats. Current efforts on threat modeling in the building energy system can be categorized into two types: modeling of passive threats and modeling of active threats. This is especially important for evaluating the robustness of fault detection and diagnosis (FDD) tools and conducting an impact analysis to understand the consequences of threats. It is more cost-effective and feasible to generate abnormal operating ground truth data using computer models than doing field experiments.

In this paper, the modeling of passive threats refers to the process of modeling passive threats that are not injected by human beings to a normal system, such as cooling coil valve stuck, sensor drifting, etc. Currently, a few works are available for the threat modeling of the building energy system especially the HVAC system. Li and O'Neill [9] gave a systematic review of physical fault modeling of HVAC system and pointed out that the number of available fault models of HVAC systems is still limited. Air handling units (AHUs), a critical HVAC equipment for conditioning and supplying air to the designated zones, were studied at a very early stage under the context of FDD. The American Society of Heating, Refrigerating and Air-Conditioning Engineers' (ASHRAE) research project RP-1312 has a specific goal of evaluating FDD tools for AHUs. A variety of fault-free and fault models (e.g., fan, valves, etc.) were developed in the HVACSIM+ platform and validated using the experimental data in ASHRAE RP-1312 [10, 11]. Cheung and Braun modeled the chiller fault models such as the refrigerant leaking, refrigerant overcharge, excess oil, non-condensable in refrigerant, condenser fouling and liquid line restriction, etc. [12, 13]. The impact of some faults such as refrigerant overcharge on the building energy consumption was numerically evaluated by integrating their fault models with a whole building simulation program of EnergyPlus [12]. Basarkar et al. [14] identified 18 common HVAC faults and simulated four faults in details using EnergyPlus. Zhang and Hong [15] extended current EnergyPlus by introducing a new EnergyPlus fault object to model and simulate various HVAC-related operational faults such as sensor faults, cooling coil fouling, dirty air filters, etc. Li and O'Neill [16] conducted 12,000 EnergyPlus fault simulations to study the fault impacts for a total of 41 faults in the medium-sized office in four different climate zones in the U.S. Khire and Trcka [17] developed a flexible and scalable Fault Manager Component in TRNSYS to enable modeling and simulation of faults in TRNSYS environment. The authors then implemented fault simulations in a commercial office building for 49 faults, which demonstrated the single fault and the 2nd-order (a combination of two faults) fault impacts. Their results showed that the fan ON at a full capacity through 24 hours has the most significant impact on the electricity demand in the summer. The demand could be as high as 60% more compared to fault-free HVAC systems. Chen et al. [18] utilized Modelica Buildings library and developed a Modelica testbed for a single duct variable air volume (VAV) system and analyzed the dynamic and steady-state impact of equipment physical faults on building operation.

The modeling of active threats with human in the middle mainly focuses on cyber-attacks launched on the communication network of a cyber-physical system. Many researchers have identified and classified vulnerabilities in the BACnet protocol, including snooping, application service attack, network layer attack, network layer Denial of Service (DoS) and application layer DoS [19, 20]. Researchers in [21, 22] have implemented a range of attacks against BACnet using the detailed specific vulnerabilities using customized BACnet simulation environments. Bowers [23] outlined a range of attacks implemented as part of an automated attack framework implemented in Python. The

framework contained three categories (i.e., discovery, enumeration and fuzzing), which utilized legitimate BACnet commands. However, researchers often forgo directly implementing an attack, rather using a synthetic data manipulation for an out-of-bounds dataset generation and testing purposes [20, 22]. Peacock identified a comprehensive list of known attacks that can be launched on a network-based control system, including DoS, flooding, smurfing, spoofing, writing attack, etc. [5]. DoS attacks are perhaps the most detrimental one that affects the packet delivery because they have been proven capable of shutting an organization off from the Internet or dramatically slowing down network links [24]. The previously mentioned researches mainly focused on the attacks and their impacts on the cyber part of a communication network system, and the building energy system in their cases is simplified or even not modeled.

In order to study the impact of cyber-attacks on a physical system, many researchers developed different attack models to disturb the physical system and observe the system's responses. The literature generally categorizes the cyber-attacks on the network-based physical system as two types: data-intrusion attack and DoS attack. Data-intrusion attack refers to the manipulation of communicated data through remote attacks that can utilize the WriteProperty to corrupt the value of the payloads. Huang et al. [25] provided basic models for data-intrusion attacks, such as max/min attack, scaling attack, additive attack, etc. Sridhar and Manimaran [26] extended the basic attacks to include ramp attack, pulse attack and random attack. Signal blocking, which is typically a byproduct of DoS attacks on the network, refers to the blocked transmissions between controllers and the plant due to the unavailability of communication devices, communication paths or local plant devices. Xin et al. [27], Sridhar and Manimaran [28] assessed the impact of signal blocking on the power grid real-time control. Sheikh et al. [29] applied machine learning techniques to classify different building operational status: normal operation, operation under faults, and operation under attacks. The attacks were modeled as signal blocking. The consequential blocked signal pattern was used to identify attacks. However, the building energy system is not modelled to evaluate the impact of cyber-attacks on the building control system.

1.2. Current Gaps

The existing threat modeling approaches for building energy systems utilize existing building performance simulators such as EnergyPlus, TRNSYS, HVACSIM+, and Modelica by either tweaking input variables in the normal model or introducing new faulty models. The following gaps are identified from the above-mentioned threat modeling efforts:

- Each single building energy simulator has a limited threat modeling and simulation capacity. EnergyPlus is the most-commonly used energy simulator for threat modeling and evaluation. However, the limitations of EnergyPlus for threat modeling are multi-fold. First, it is difficult to model control-related faults in EnergyPlus, such as inappropriate PID parameters and communication delays, due to its assumption of ideal controllers. Second, the injection of pressure-involved faults is not straightforward. For example, to model duct fouling, users need to adjust the pressure head, minimum and maximum air flowrate in the fan model. Third, the EnergyPlus *FaultModel* object has limited threat modeling capacity, and needs to be expanded to include more fault scenarios such as water temperature sensor faults, etc. Modelica is an object-oriented modeling language, and is able to provide some levels of the modeling flexibility. Current research that utilizes Modelica for threat modeling and simulation in a building energy and control system exposes several limitations as well. First, the threat-free building energy and control system is typically built on an open-source Modelica Buildings library [30]. Modelica Buildings library is mainly designed to support the modeling and simulation of the normal buildings operation. Efforts are still required to extend this library to support threat modeling. Second, existing approaches, such as in [18], for injecting threats are not generalizable. They are either demonstrated using a specific simulation tool or incompletely defined. For example, it would be impractical to model active threats, such as cyber-attacks, without specifying their start and end times.

- The integration of active threat modeling with building energy simulators are imperative for GEBs. However, there is barely a framework that considers the modeling of active threats, let alone the integration of both passive and active threats in a single and scalable platform. EnergyPlus is mostly used to support passive threats modeling and simulation. The large simulation time step (at least 1 minute) and assumption of ideal controller make EnerguPlus insufficient to model and simulate control-related threats, especially short-term cyber-attacks launched on a network-based control system.

- There lacks a flexible and generic threat modeling and simulation framework. First, current research defines their own threats differently in different platforms, which makes them difficult to be reused. A standard and scalable library for threat definition and injection is needed. This library should be able to define a threat or threats in a well-structured platform, and provide the capability of injecting such threats to a threat-free system as defined in the same platform or other platforms. Second, current threat injections are mostly performed in a single simulator, which inherits the limited capacity of such a simulator. For example, EnergyPlus provides limited support for injecting threats at an arbitrary location, from a random start time and for any threat duration. Although Modelica-based simulator can be flexible but the current research failed to present a more generic threat modeling framework based on Modelica.

This paper presents a flexible and generic functional mock-up unit (FMU) based threat injection framework in Python. FMU is a simulation model that is compliant to the functional mock-up interface (FMI), which defines a standardized interface for a model so that such a model can be simulated in a different environment for model exchange or co-simulation. The utilization of FMU in this paper is to provide a standardized model interface for different building energy simulators so that a generic threat library can be connected to it and perform the threat evaluation. The proposed framework is built on existing energy simulators, such as Modelica, but is not limited to be used with Modelica. The model fidelity is usually defined by the energy simulators, not this framework. Currently, EnergyPlus, TRNSYS and Modelica models can all be compiled to FMU models. Therefore, this proposed framework can be reused for all three simulators but needs minimum modifications to consider their different software structures. For example, the required modifications for extending this framework for EnergyPlus are shortly discussed in the future work of this paper. The highlights of this research are summarized as below:

- This research provides a flexible and generic threat-injection framework for any FMU-based simulators.
- This framework provides a generic definition of threat and threat injection.
- This generic threat injection framework can model and simulate both passive threats and active threats by leveraging the Modelica simulator.
- This research demonstrates the capability and effectiveness of the threat-injection framework on a Modelica/FMU-based building energy and control system.

The rest of this paper is organized as follows. Section 2 introduces the proposed threat injection framework based on the Modelica simulator. Section 3 numerically demonstrates the capability of this framework. Different orders of threats are modelled and simulated, followed by final conclusions in Section 4.

2. Proposed Threat Injection Framework

The software architecture of this framework is illustrated in Figure 1. A threat-free template model based on a normal threat-free Modelica model is first created in a Modelica environment using a developed Modelica *Overwrite* package. The template model is then parsed and used to auto-generate a standalone FMU wrapper model (*wrapper.mo*), which is structured for threat injection. Meanwhile, different types of threats are defined in a developed Python library (*pyThreat*). These defined threats are then injected into the FMU wrapper model by utilizing the existing *pyFMI* library and a developed *pyThreat* library. The threat-injected model is simulated in a simulation manager developed as the Python *pySimulate* library. The FMUs in the above modules are based on FMI 2.0. Although FMI 2.0 provides two interface types (e.g., co-simulation and model exchange), we only tested our framework on the co-simulation interface to allow for a future extension of the framework to EnergyPlus, which currently supports a co-simulation FMU only [31]. The summary of each module as in Figure 1 is listed as below:

Threat-free Modelica Template. This module defines a threat-free Modelica model based on a normal Modelica system or a component model by using the developed Modelica *Overwrite* package. The Modelica template model defines the status and the location of a set of threat injection actions. For example, if a chiller is to be maliciously controlled off during a peak-load period, the on/off control signal received by the chiller should be overwritten by

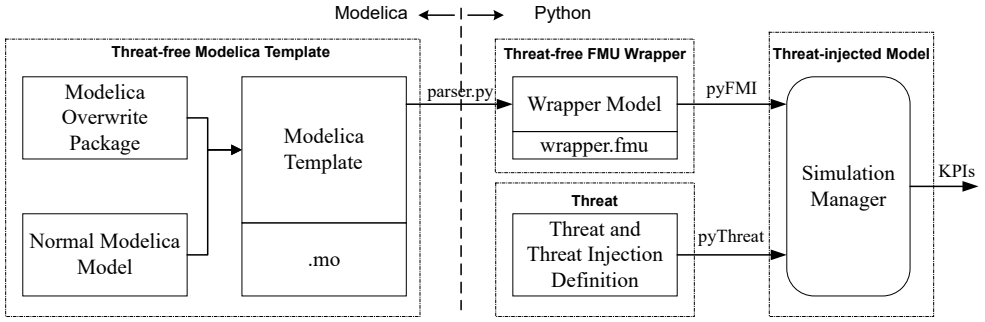

Figure 1: Schematic of threat injection framework

external programs (such as Python in this framework). The *Overwrite* package should be used here to make the injection of chiller on/off signal ready. This template should be simulator-specific when the same framework is applied to other simulators such as EnergyPlus and TRNSYS. For example, a similar threat-free template may be created for EnergyPlus through its built-in energy management system (EMS) module that enables EnergyPlus receive external signals. Recently released Python EMS [32] can be utilized to facilitate creating such a template in EnergyPlus.

Threat-free FMU Wrapper. The Modelica template is then parsed to auto-generate a FMU wrapper model, which wraps the template model by an extra input layer that defines all the possible threat locations (Modelica input connectors). That is, the wrapper model exposes all the threat injection locations defined by the instances of *Overwrite* package in the template model. These injection locations (Modelica input connectors) can then be used to receive external threats. This part also needs some limited modifications if used for non-Modelica simulators by considering simulator-specific syntax for code auto-generation.

Threat. Threats are defined in a Python environment using the *pyThreat* library, which can be reused by any other non-Modelica simulator. A threat is constructed by assigning key attributes such as start time, end time, injection location and type, temporal pattern, etc.

Simulation Manager. Simulation manager defines a simulation case by integrating Python threats with the FMU wrapper, and performs the numerical simulation. The simulation results are processed by outputting real-time key performance indexes (KPIs).

2.1. Threat-free Modelica Template

Modelica is an open-source object-oriented modeling language [33], and has emerged as a dynamic modeling tool for building energy and control systems [30]. Currently, Modelica Buildings library developed by the U.S. Department of Energy Lawrence Berkeley National Laboratory has been widely used for the design and control of building energy system [30, 34, 35, 36, 37, 38]. However, Modelica Buildings library is designed to evaluate the threat-free energy and control system. Efforts are stilled needed to expand this library for threat modeling.

In this framework, a Modelica package, namely *Overwrite*, is developed and used to provide the flexibility for overwriting the normal system operation status (e.g., measurements, control signals, etc.) from external sources. This package is highly inspired by [39]. Figure 2 illustrates how to build a threat-free Modelica template from a normal Modelica model and the proposed *Overwrite* package. The normal Modelica model simulates a building energy and control system under norm operations, where a control signal u is generated from a supervisory-level control system and sent to the local plant. This is to mimic the network-based control system in real buildings. Then the *overwrite* instances from the *Overwrite* package are injected in the normal Modelica model to create a threat-free Modelica template. The *overwrite* instance serves as a switch between a normal signal (u_1) and a threatened signal (u_2). The number of *overwrite* instances in a template depends on how many variables a user wants to overwrite. In the template, the overwrite instances are not activated as default. Therefore, the original normal signal is passed through the *overwrite* instance in the template. The template is then used to generate a Modelica/FMU wrapper by identifying those to-be-overwritten variables through the *overwrite* instances, and creating two external interfaces

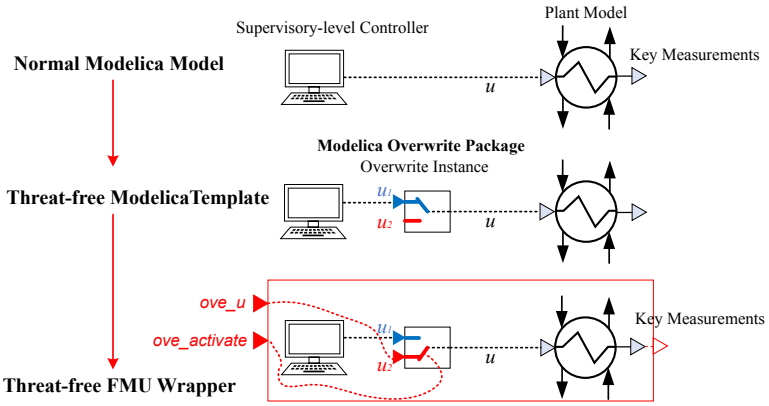

Figure 2: Schematic process for generating the threat-free FMU wrapper

for each overwrite instance to receive external signals. For instance, *ove_u* is an interface for receiving an external threatened signal and connected to u_2 in the *overwrite* instance, and *ove_activate* is an interface to indicate if *ove_u* should be activated to overwrite an original normal signal. More details about how the data flow is achieved are presented in Section 2.1.1.

2.1.1. Modelica Overwrite Package

Two basic models are introduced in this section to overwrite a Modelica variable and a Modelica parameter during the simulation from external sources. These two models are named *OverwriteVariable*, and *OverwriteParameter*, respectively. The detailed implementations are explained as follows.

OverwriteVariable. This model passes an external signal instead of original input signal when the overwriting action is activated. This model is typically used by the threat injection framework to identify and activate signals that can be overwritten by external threats.

OverwriteParameter. This model is used when there is at least one Modelica parameter to be overwritten during a simulation. The parameter p in this model intends to be overwritten by the external threat injection library between time steps, and has no physical meaning until being assigned to a threatened parameter of a model. For instance, a typical parameter of the cooling coil model is known as nominal UA. The nominal UA should not be changed within a simulation experiment when there is no fouling. However, when the cooling coil is gradually fouled, the heat resistance increases as well, and thus the nominal UA decreases over time. To model this effect, *OverwriteParameter* can be used at the top level of the Modelica model by assigning p to the nominal UA of the cooling coil model. In this way, any changes on p will be directly propagated to the nominal UA.

Figure 3 illustrates the use of *Overwrite* package to create a Modelica template that is then used to generate a wrapper model with overwritten signals exposed to external programs such as attackers as defined in Python. This template describes a HVAC and its control system, including a primary chilled water loop, a condenser water loop and a single-duct VAV system serving five thermal zones as described in Section 3.

The template uses an instance of *OverwriteVariable* named *oveTCHWSet* to overwrite the chilled water supply temperature setpoint, and an instance of *OverwriteParameter* named *oveTCooOn* to overwrite the zone air temperature setpoint when the cooling is on. These instances are then used to generate a wrapper for the Modelica template so that all the parameters and variables that are to be overwritten are exposed to external attackers. During the normal operation, the overwriting is not activated. When threats are injected, these instances are activated to take external control signals. More details about the wrapper can be found at Section 2.2.

2.2. Threat-free FMU Wrapper

This section describes the automatic construction of a wrapper model based on the given Modelica template. Two types of signals can exchange the data with external entities as defined in the Modelica template: Modelica variable

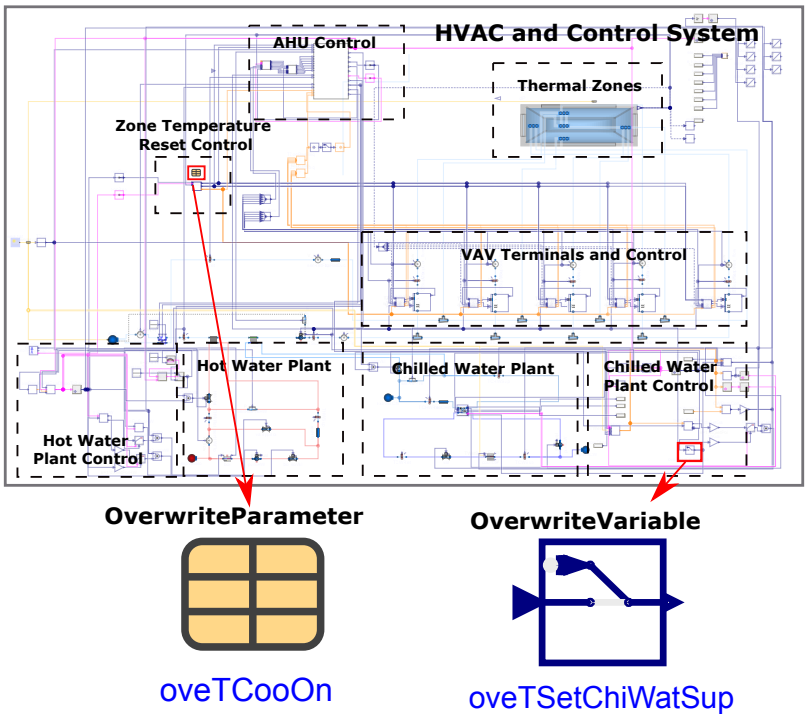

Figure 3: An example of Modelica template with *Overwrite* instances

and Modelica parameter. Although the declaration of a Modelica variable or a Modelica parameter depends on the needs of the model, a Modelica variable is usually time-variant and represents control inputs and system measurements, and a Modelica parameter is usually time-invariant and used for system high-level configuration settings such as equipment sizes. With the utilization of the *Overwrite* package as mentioned in Section 2.1.1, the FMU model of the Modelica template can be automatically parsed using a Python library *parsing* to locate the instances of *OverwriteVariable* and/or *OverwriteParameter*. These instances are then used to create a Modelica/FMU wrapper for threat injection and simulation in a Python environment.

A *parsing* library is developed in Python and contains the following main functions:

- Translate the Modelica template into a FMU template. The python package *pyDymola* that inherits *BuildingsPy* [40] is developed to generate Dymola FMU, and the python library *pymodelica* that was developed by Modelon is used here to generate a jModelica FMU [41].

- Locate the exchanged signals in the FMU template. This library parses the FMU documentation, and locates the exchanged signals by searching the keywords as defined in the *Overwrite* Modelica package. All the instances of *OverwriteVariable* and *OverwriteParameter* are located in this step.

- Write a Modelica wrapper by propagating exchanged signals to the top-level. The python package *parser* automatically writes a Modelica wrapper model using the information of located instances. First, a Modelica wrapper model named *wrapper.mo* is created by instantiating an instance of the original Modelica template. Second, two inputs for each instance of *OverwriteVariable* are added to the wrapper. The inputs are named *block_instance_u* and *block_instance_activate*. The unit, description, minimum and maximum value, start value and other signal attributes are read and assigned to each *block_instance_u*. Third, one parameter for each instance of *OverwriteParameter* is added to the wrapper. The parameter is named as *block_instance_p*. The unit, description, minimum and maximum value, start value and other signal attributes are read and assigned to each *block_instance_p* as well. Forth, the parser automatically connects *block_instance_u* to *block.instance.u* and

block_instance_activate to *block.instance.activate*. Fifth, the parser automatically connects *block_instance_p* to *block.instance.p*.

- Translate the Modelica wrapper model into a FMU wrapper model. The FMU wrapper is named *wrapper.fmu*.

An external interface may use the inputs and parameters in the wrapper model to send corrupted signals to specific overwrite blocks and activation signals (activate) to enable and disable signal overwriting. By default, the activation of the signal overwrite block is set to *False*. In this way, external interfaces need to only define control signals for those that are being overwritten.

Figure 4 shows a generated Modelica wrapper for the Modelica template as illustrated in Figure 3. The template contains one instance of *OverwriteVariable* named *oveTSetChiWatSup* to overwrite the normal chilled water supply temperature setpoint, and instance of *OverwriteParameter* named *oveTCooOn* to overwrite the global zone air temperature setpoint when the cooling is on. The generated wrapper instantiates the original Modelica template to represent an HVAC and control system under normal operations. Two inputs *oveTSetChiWatSup_u* and *oveTSetChiWatSup_activate* are added to receive external chilled water supply temperature setpoints and the activation status of overwriting. One parameter *oveTCooOn_p* is added to the wrapper to receive external cooling setpoints for zones.

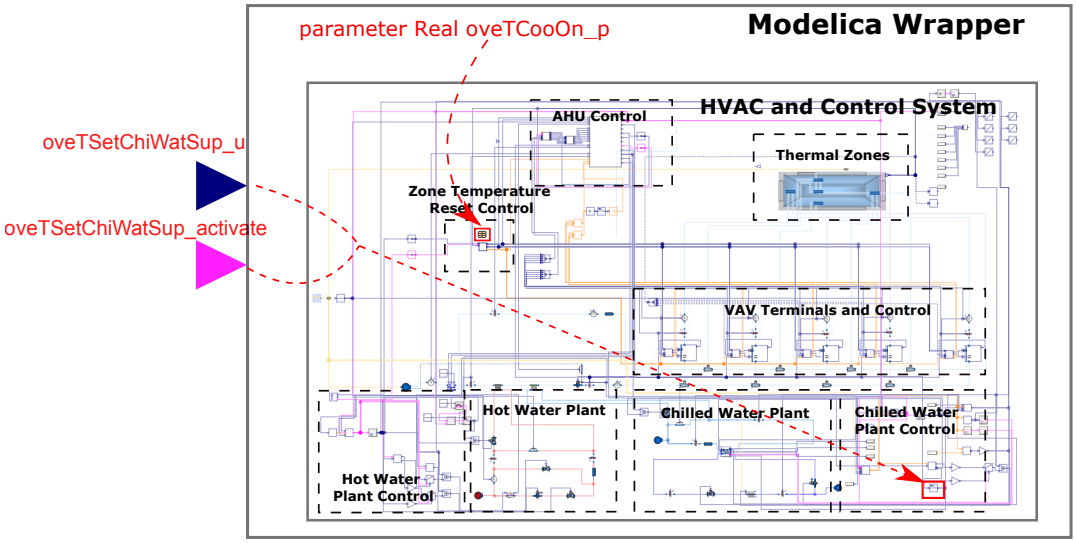

Figure 4: An example of auto-generated Modelica wrapper

2.3. Threat and Threat Injection Library

2.3.1. Threat Definition

This part introduces a Python library *pyThreat* that defines a threat and its injection scenario. This library is a standalone library, and can be used for any energy simulators that support FMU exporting. A threat as defined in this library has the following attributes:

- *active* Flag of threat status. If a threat is active, then the signals required to exchange the data with external entities are exposed in the wrapper model. If a threat is inactive, then the signals of the threat are not exposed in the wrapper model.
- *start_threat* Start time of a threat. If a threat is active, then it is injected to the wrapper model at this start time.
- *end_threat* End time of a threat. If a threat is active, then it is injected to the wrapper model till this end time.
- *temporal* Temporal pattern of a threat. This attribute defines how a threat evolves over time. Multiple basic temporal models are provided in the library, including *max*, *constant*, *pulse*, *blocking*, etc. These models are detailed in the rest of this section.

- *injection* Injection type. This attribute defines how a threat is injected into a Modelica model. Two types of injection types are available: 'p' is for the time-invariant parameter type injection and 'v' is for the time-variant variable type injection. The selection of these two types depends on how the threatened signal is modeled in Modelica/FMU.

The *temporal* defines the trajectory of the threatened signal. It could be used to simulate a passive threat, for example, fouling cooling coils, and represent an attack behavior that changes control settings or sensor readings of a targeted system. The *temporal* is implemented as a Python class, and any instances of such a class can be used to define a specific temporal pattern and be used for a threat definition. The class *temporal* itself contains two attributes. One is the type of the signal to be overwritten, specifying the name in the Modelica template model, minimum and maximum value, and its unit. The other is the temporal patterns of a signal under threat.

The following basic temporal models are considered in this study. Users can also customize their own temporal models and integrate them in this *pyThreat* library. In the following equations, $\hat{y}$ is the corrupted property value, y is the original value, $\mathbb{A}$ is the threat period, the subscripts *min* and *max* represent the minimum and maximum value of a signal, respectively. The minimum and maximum values are needed to mimic the fact that a typical building automation system would generate warnings and self-correct the values when deviations from bounded range are detected.

Max. The property value is overwritten to its maximum allowable value during the threat.

$$\hat{y}(t) = \begin{cases} y(t), & t \notin \mathbb{A} \\ y_{max}, & t \in \mathbb{A} \end{cases} \tag{1}$$

Constant. The property value is overwritten to a user-defined constant value k during the threat.

$$\hat{y}(t) = \begin{cases} y(t), & t \notin \mathbb{A} \\ k, & t \in \mathbb{A} \text{ and } k \in [y_{min}, y_{max}] \\ y_{min}, & t \in \mathbb{A} \text{ and } k < y_{min} \\ y_{max}, & t \in \mathbb{A} \text{ and } k > y_{max} \end{cases} \tag{2}$$

Pulse. A pulse attack involves modifying true values through a pulse wave function $f_p(t)$.

$$\hat{y}(t) = \begin{cases} y(t), & t \notin \mathbb{A} \\ f_p(t), & t \in \mathbb{A} \text{ and } f_p(t) \in [y_{min}, y_{max}] \\ y_{min}, & t \in \mathbb{A} \text{ and } f_p(t) < y_{min} \\ y_{max}, & t \in \mathbb{A} \text{ and } f_p(t) > y_{max} \end{cases} \tag{3}$$

Blocking. A Denial-of-service (DoS) attack targeted at a control system aims at either making the sensor measurements unavailable to the control center or the control signal unavailable to the physical system in time. The unavailability takes the format of signal blocking, where the downstream receiver cannot receive any signals in this situation and will have to use values from the previous time step.

$$\hat{y}(t) = \begin{cases} y(t), & t \notin \mathbb{A} \\ y(t-1), & t \in \mathbb{A} \end{cases} \tag{4}$$

2.3.2. Threat Injection

The threats as defined in the previous section can be injected into a Modelica model by two ways depending on how the threatened signals are defined in the model. These two injection methods are variable injection ("v") and parameter injection ("p"). The injection type is defined based on the Modelica syntax. Modelica parameters and variables are handled differently after the compilation. The Modelica parameters are typically defined at the top-level of a model, and difficult to change in Modelica after the compilation, while Modelica variable is at the low-level and time-variant, and could be changed over time. The detailed explanation about parameter- and variable-type injections are illustrated as below.

Time-invariant Parameter Injection. Time-invariant variable in Modelica is known as constant that never changes but can be substituted by its value. An example is the gravitational acceleration constant g on the Earth that is 9.8 m/s^2. A special type of time-invariant variables is a simulation parameter. Values of these simulation parameter constants are assigned only at the start of the simulation and kept as the constant during a simulation. For example, the nominal capacity of a chiller is a simulation parameter, and it will not change during a simulation. Time-invariant Parameter Injection means a threat is injected to a time-invariant parameter by overwriting its value during a simulations if possible. This type of threat injection applies to scenarios such as fouling cooling coils with a degraded nominal UA, or degraded fan efficiency. However, within one simulation, the value of a simulation parameter cannot be changed as defined in Modelica. One workaround is to split a simulation into a group of consequential sub-simulations with a shorter duration, and assign different values for the same parameter at each sub-simulation. For example, a one-day simulation can be split into 24 one-hour consequential simulations. And for each short-simulation, the parameter can be assigned to different values based on threat injection needs.

Time-variant Variable Injection. Time-variant Variable in Modelica refers to a kind of Modelica object that can change its value over time during a simulation. An example is the output of a controller, which is adaptively updated over time to respond to external disturbances. Time-variant Variable Injection means threats are injected to time-variant variables by overwriting its value during a simulation. This type of injection applies to the majority of HVAC equipment and system faults such as stuck actuators, sensor drifting, etc.

2.4. Simulation Manager

Simulation manager defines a Python class *case* for different simulation scenarios. This library aims to provide the capability of simulating a co-simulation FMU model with external inputs from the Python environment. The simulation advances one experiment step until reaches the end of the experiment. Within each experiment step, the simulation manager decides if the current step should be divided into multiple smaller steps based on the needs of changing Modelica parameters during a simulation, such as injecting a parameter-type threat in the middle of current experiment step. The simulation results of key measurements are saved for the post-processing.

This library contains three major methods to perform co-simulation between Python and FMU:

baseline: This method is to calculate the baseline response of such a system without any threats injected. The calculation lasts from the experiment start time to the experiment end time.

simulate: This method is designed for simulating the threat-injected system. It first parses the wrapper FMU model and check if the names of threatened variables are correctly exposed, then parses a threat list to extract active threats, and categorize them into variable-type and parameter-type based on their injection types. The simulation is then performed for each time step using the *do_step* method.

do_step: This method is to perform a step-wise co-simulation between the FMU wrapper model and the threat injection model as defined in Python. Smaller steps are required if during the current step any parameter-type threats and variable-type threats are injected. Current co-simulation step is then divided into smaller steps as illustrated in Figure 5. For example, at the step t_{n-1}, two threats (th_1 and th_2) are injected. The threat th_1 ends within the current step, while the threat th_2 does not end in the current step. To be able to change the values of FMU parameters and inputs for each threat, the start and end time of each threat is used to divide current co-simulation time step into smaller steps (e.g., $\Delta t_1, \Delta t_2, \Delta t_3$, and Δt_4). In each small step, the values of threatened variables or parameters are accordingly updated.

2.5. Key Performance Indexes

Key Performance Indexes (KPIs) are used to quantify the effect of threats on the building energy and control systems. For this study of GEB, the KPIs are categorized into two types: quality of building service and quality of grid service, and discussed as follows.

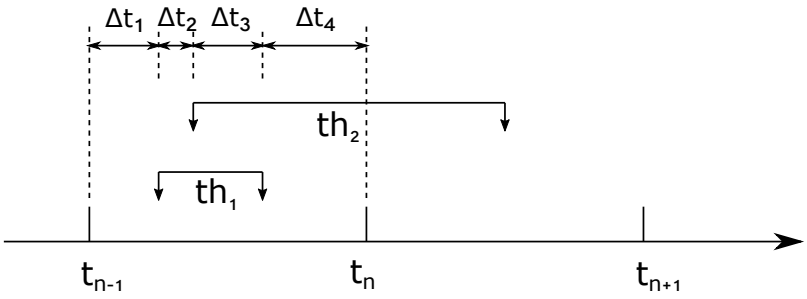

Figure 5: Illustration of dividing a communication step into smaller steps

2.5.1. Quality of Building Service

Buildings are designed to deliver specific services to the occupants. For instance, a commercial building usually provides thermal, visual and acoustic comfort to its occupants by consuming energy. This study utilizes a list of KPIs to evaluate how the building service would be compromised under threats. The KPIs include energy efficiency, power demand, and thermal comfort, etc. Details about the calculation are introduced in the following:

Energy Usage. Energy efficiency for a specific building can be quantified by its energy usage defined as below.

$$E = \int_{t_s}^{t_e} P(t)dt \tag{5}$$

where E denotes the energy usage in the system during the studied period, t_s denotes the start time of an experiment, t_e denotes the end time of an experiment and $P(t)$ denotes the power demand of the system at time t.

Peak Power Demand. The peak power during a time period is calculated as:

$$P_{max} = \max_{t_s \le t \le t_e} (P(t)) \tag{6}$$

Thermal Discomfort. The thermal discomfort is the integral of the deviation of the temperature with respect to a predefined comfort range during a given period of time. The units are Kh and is quantified as:

$$D = \sum_{z \in \mathbb{Z}} \int_{t_s}^{t_e} \|d_z(t)\| \, dt \tag{7}$$

where D is the total discomfort between the initial time t_s and the final time t_e; z is the zone index for the set of zones in the building $\mathbb{Z}$; $d_z(t)$ is the temperature deviation in K from the lower and upper setpoint temperatures established in zone z.

2.5.2. Quality of Grid Service

Demand Flexibility. Demand flexibility describes how flexible a building can change its power demand during a time duration. A threat imposed on the building energy and control system might compromise the building's original flexibility and thus impact the amount and quality of grid services it can provide.

A demand flexibility indicator (DFI) specifies the amount of power variation, over a fixed duration, that is available from a building. A building DFI is time varying and depends on the system states at time t. The detailed calculation of DFI used in this paper is as follows [42].

$$\overline{DFI}(t, \Delta t) = \frac{\int_t^{t+\Delta t} \overline{P} dt - \int_t^{t+\Delta t} P_{ref} dt}{\Delta t}, t \in [t_s, t_e) \tag{8}$$

$$\underline{DFI}(t, \Delta t) = \frac{\int_t^{t+\Delta t} P_{ref} dt - \int_t^{t+\Delta t} \underline{P} dt}{\Delta t}, t \in [t_s, t_e) \tag{9}$$

where Δt is the duration of the flexibility, DFI indicates demand flexibility, and its actual value is an average over the duration period. P_{ref} is the reference power trajectory when not providing grid services. $\overline{P}$ and $\underline{P}$ are the upward and downward power trajectory when providing grid service. The upward and downward flexible power ($\overline{P}$, $\underline{P}$) at each time t can be achieved by grid service delivery strategies such as increasing zone air temperature setpoint during cooling for upward flexibility. This paper uses the following strategy to achieve downward and upward flexibility: increase zone air temperature setpoint by 2 °C to achieve downward power flexibility and decrease the zone temperature by 2 °C to gain upward flexibility. Note that different strategies can be used to harvest the demand flexibility in buildings.

3. Case Study

This section introduces a case study to demonstrate the proposed framework. Section 3.1 describes the studied baseline HVAC and its control system. Section 3.2 designs the different threats and different threat-injection scenarios, followed by detailed result discussions in the remaining subsections.

3.1. Baseline System and Validation

Figure 6 is a schematic drawing of a typical HVAC system for a medium-sized office building. Heating and cooling are delivered by a single-duct VAV system. One AHU connected with five VAV terminal boxes serves five zones (four exterior zones and one interior zone) on one floor. The chilled water is supplied by a central chiller plant which consists of a chiller, a waterside economizer, a cooling tower, one chilled water pump, and one cooling water pump. A boiler, powered by natural gas, supplies the hot water to the AHU heating coils. The office is occupied from 6am to 7pm on weekdays. The baseline system is sized and validated against the DOE prototype medium-sized office building as developed in EnergyPlus [43].

The HVAC system control complies with ASHRAE guidelines or literature-reported practices. For example, the air loop uses rule-based supervisory control logic recommended from ASHRAE Guideline 36 [44, 45], and the water loop conforms with ASHRAE RP-1711 [37, 46].

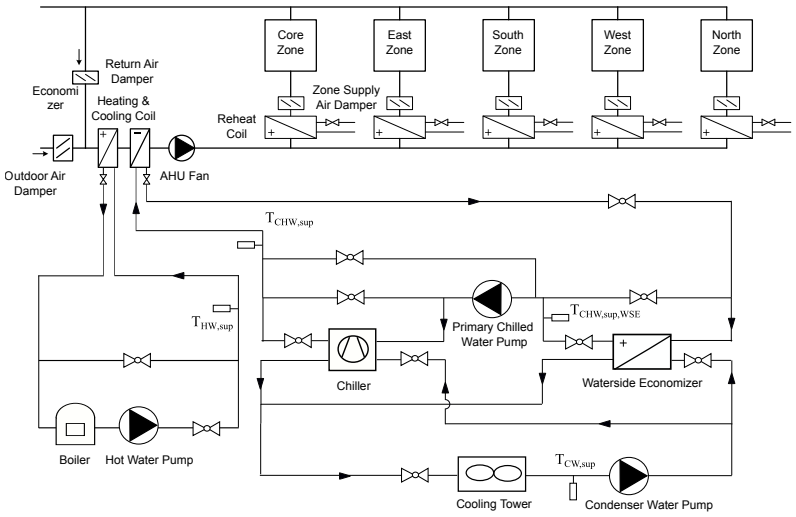

Figure 6: Schematics of a typical HVAC system

3.2. Cases Design

In this section, four threats are defined and simulated, as summarized in Table 1. Threat 1 refers to a malicious change of the fan speed control command at the supervisory level. Threat 2 manipulates the on/off signal of a chiller to represent a remote cyber-attack. Threat 3 blocks the chiller from receiving chilled water supply temperature setpoint

from the supervisory controller. Threat 4 maliciously lowers the global zone air temperature cooling setpoint so that more power would be used by the system. The four threats are injected during on-peak period to assess their consequences of both building service and grid service. By combining these four threats with different active status, we obtained 15 simulation cases. Table 2 summarizes the detailed active status of each threat for each case, where the *Order* refers to the number of simultaneously active threats in a simulation. All cases are simulated for a hot summer day, when the chiller is activated to provide cooling to the building.

Table 1: Summary of injected threats

Threat	Threat Type	Location	Start Time	End Time	Injection Type	Temporal	Description
1	Data-intrusion attack	supply air fan speed	7am	7pm	v[1]	max	maliciously change supply air fan speed to its maximum value
2	Data-intrusion attack	chiller on/off signal	12pm	3pm	v	square pulse	launch a square pulse attack with a period of one hour on chiller on/off control signal
3	DoS attack	chilled water supply temperature setpoint	10am	1pm	v	block	block chiller from receiving chilled water temperature setpoint
4	Data-intrusion attack	zone temperature cooling setpoint	1pm	3pm	p[2]	constant	change zone temperature cooling setpoint to a constant value of 22 °C

[1]:time-variant variable injection

[2]:time-invariant parameter injection

Table 2: Summary of numerical cases

Case	Order	Threat 1	Threat 2	Threat 3	Threat 4
1	1	✓			
2	1		✓		
3	1			✓	
4	1				✓
5	2	✓	✓		
6	2	✓		✓	
7	2	✓			✓
8	2		✓	✓	
9	2		✓		✓
10	2			✓	✓
11	3	✓	✓	✓	
12	3	✓	✓		✓
13	3	✓		✓	✓
14	3		✓	✓	✓
15	4	✓	✓	✓	✓

3.3. First-order Threat Injection

3.3.1. Case 1

This scenario simulates an active threat that operates the supply air fan at its maximum speed during the occupied period. The threat is injected on a typical summer day. Detailed system responses in terms of system states and power consumption are shown in Figure 7 and Figure 8, respectively.

At the beginning of the occupied period (6am-8:30am) when the building cooling load is small, all VAV terminal dampers open at the minimum position. The supply fan in the baseline system operates around its minimum speed to maintain the room air temperature at its setpoint. When the fan is attacked to operate at its full speed during this period, based on the fan curve law, both air flow rate and supply fan differential pressure increase. This introduces extra supply air to over-cool each zone, which leads to zone air temperatures lower than their setpoints. This over-cooling phenomena in the morning introduces additional cooling load to the chiller compared with that in the baseline system. To deal with the additional cooling load, the supervisory controller, which uses trim and respond schemes to reset the differential pressure setpoint and supply water temperature setpoint in the chilled water loop, raises differential pressure setpoint gradually to the maximum value to deliver more chilled water through the cooling coil and cool the mixed air. When the differential pressure setpoint is at its maximum allowed value and the supply air temperature is still above its setpoint, then the chilled water supply temperature setpoint is lowered gradually from its maximum value as shown in Figure 7(b)(c). The extra cooling load in the early morning due to the threat results in more power use compared with that in the baseline system as illustrated in Figure 8.

When the cooling load in the buildings increases in the afternoon, although the supply fan operates at a higher speed than that in the baseline system, the system states for these two systems are very similar. The similarity is due to the interactive control among different components within such a system. For example, when the cooling load is large and the VAV damper needs to open more than its minimum position, the influence of the increase of the fan speed on the zone temperature can be mitigated by reducing the VAV damper opening, which can maintain a similar air flow rate in the zone. Such self-healing behaviors in the building HAVC control system provides some resilience under the threat. However, although the system states during the peak period are not significantly influenced, the full speed fan consumes much more power than that in the baseline as shown in Figure 8(b). The significant amount of total power rise in the threatened system comes from the threatened supply fan controller.

3.3.2. Case 2

This scenario simulates a threat that corrupts the chiller ON/OFF control signal sent from the management level to the local chiller. The threat is injected from 12pm to 3pm on a typical day in the cooling season. The attacker injects a pulse wave signal to the chiller so that the chiller cycles ON/OFF every 30 minutes.

During the threat, the frequent activation and deactivation of the chiller leads to the oscillating system states. When the chiller is off for 30 minutes, the zone air temperature and AHU supply air temperature (SAT) increase due to the system's inability to provide sufficient chilled water. To meet the ventilation requirement in the zones, the high temperature outdoor air is mixed with the zone return air, which leads to a higher SAT than the zone temperature when the cooling is unavailable. The VAV damper then opens at its minimum position to only supply the minimum air to meet the zone's ventilation needs. Thus, the supply fan only needs to run at a small speed to provide the required minimum air flow rate. When the chiller is on for 30 minutes, all the equipment work harder to recover the deviated system states to normal conditions as soon as possible. However, within 30 minutes, the system cannot be fully recovered. For example, the SAT cannot be cooled down to its setpoint in 30 minutes when the chiller is on during the pulse attack as shown in Figure 9(c). This eventually causes escalating SAT deviations from its setpoint over the pulse attack period. The same escalating deviations also can be observed in the zone air temperatures, VAV damper openings, and supply fan speed, etc. The oscillating system states consequentially result in the oscillating system power as shown in Figure 10.

During the post-threat period (3pm-7pm), the system spends significant efforts to recover from the deviations. As shown in Figure 9(g)(i), the controller sets the chiller water supply temperature (CHWST) to its minimum value to maximize the provision of cooling, the supply air fan operates at a high speed to blow the cold air into the zones as much as it could. This eventually leads to significant power consumption of each major component as shown in Figure 10. The system almost operates at its full capacity at the beginning of the post-attack, and gradually decreases its output as the deviations of the system states decrease. In addition, different system states require different recovery times due to their specific inertia and the control system designs. For example, the zone air temperature requires about 4 hours to get back to its setpoint, while the SAT only needs about 1.5 hours. All these observations can guide the design of threat mitigation techniques to avoid high power peak or large energy consumption after the pulse attack.

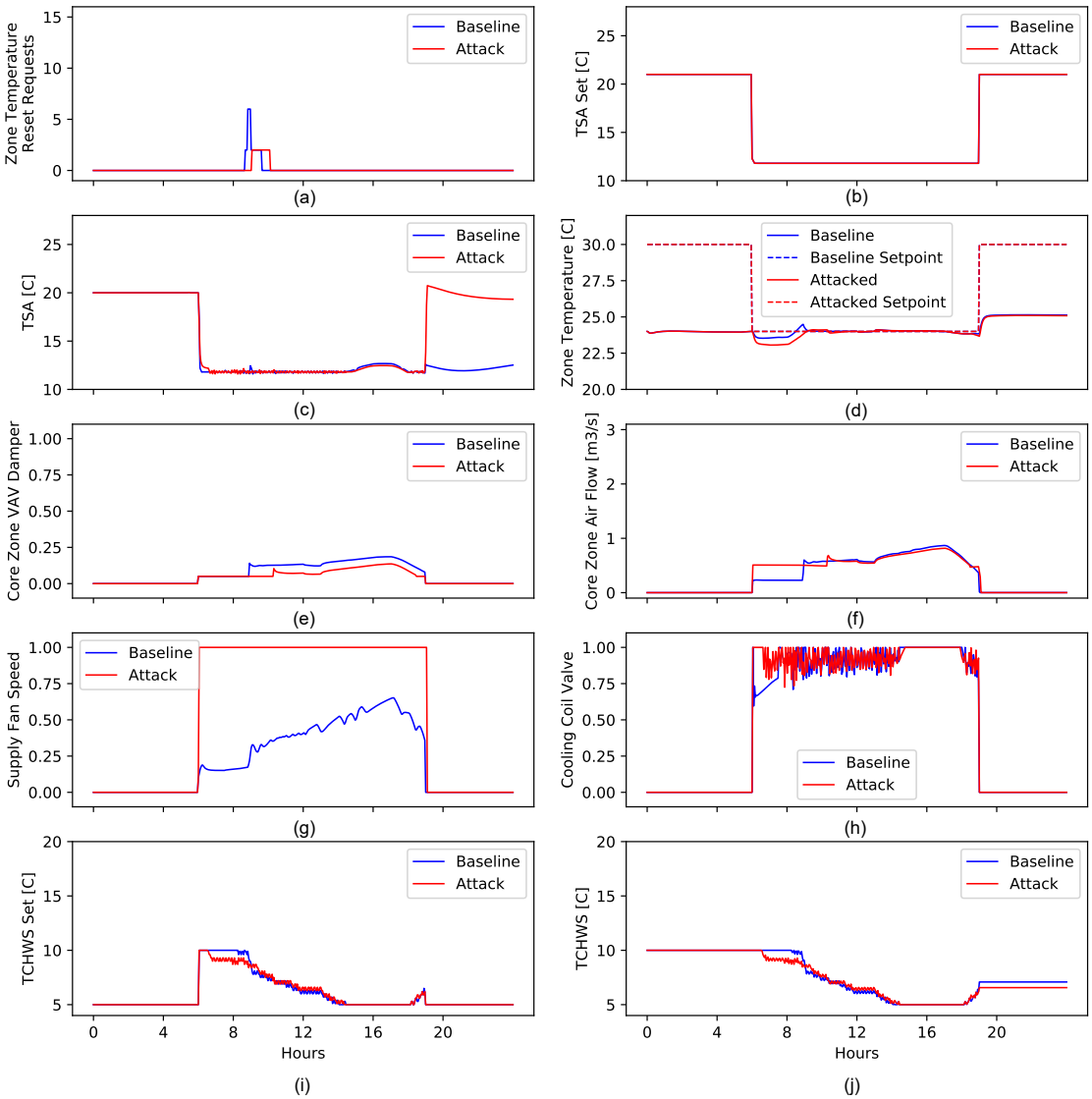

Figure 7: System states comparison between case 1 and baseline

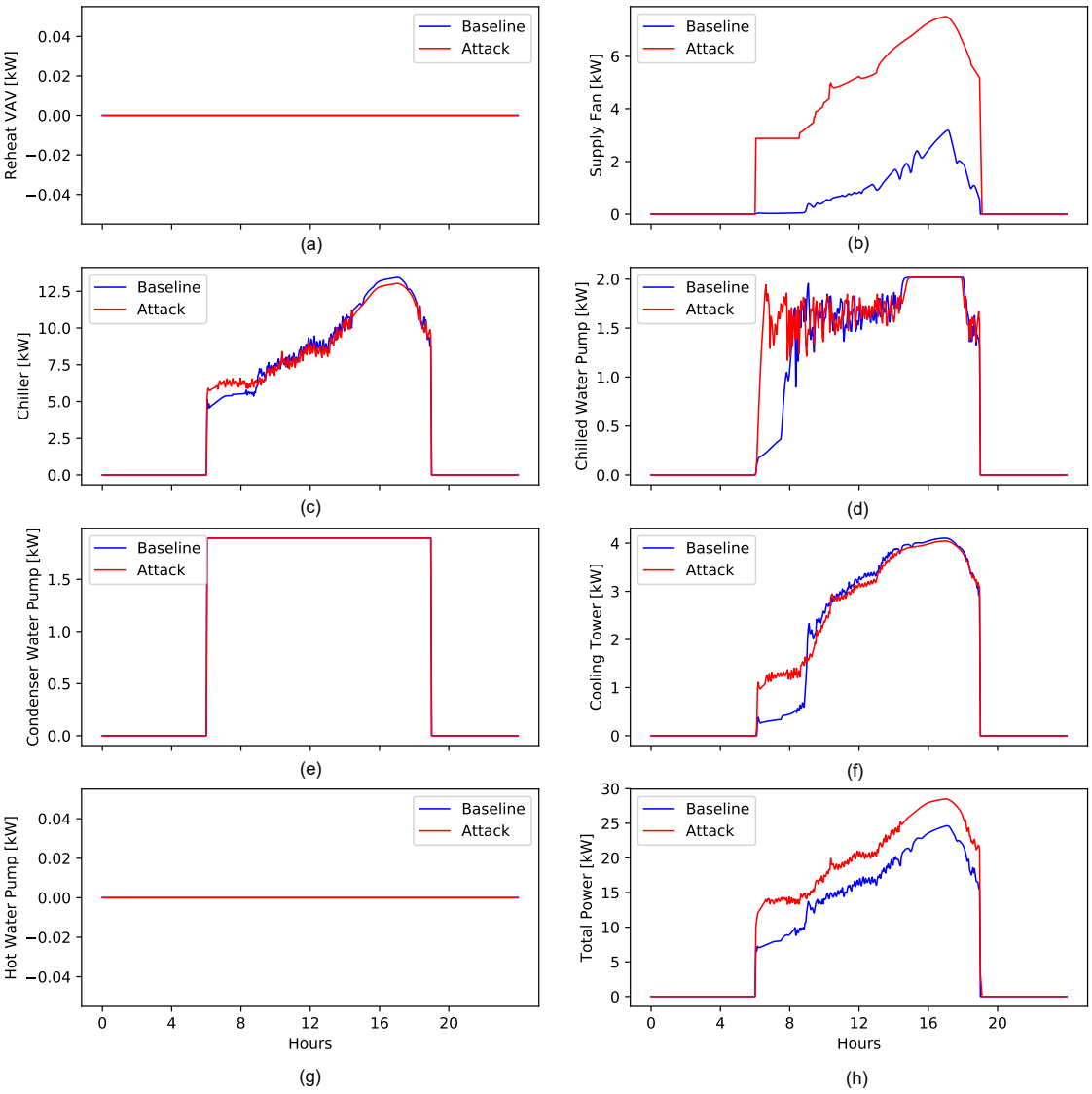

Figure 8: System power comparison between case 1 and baseline

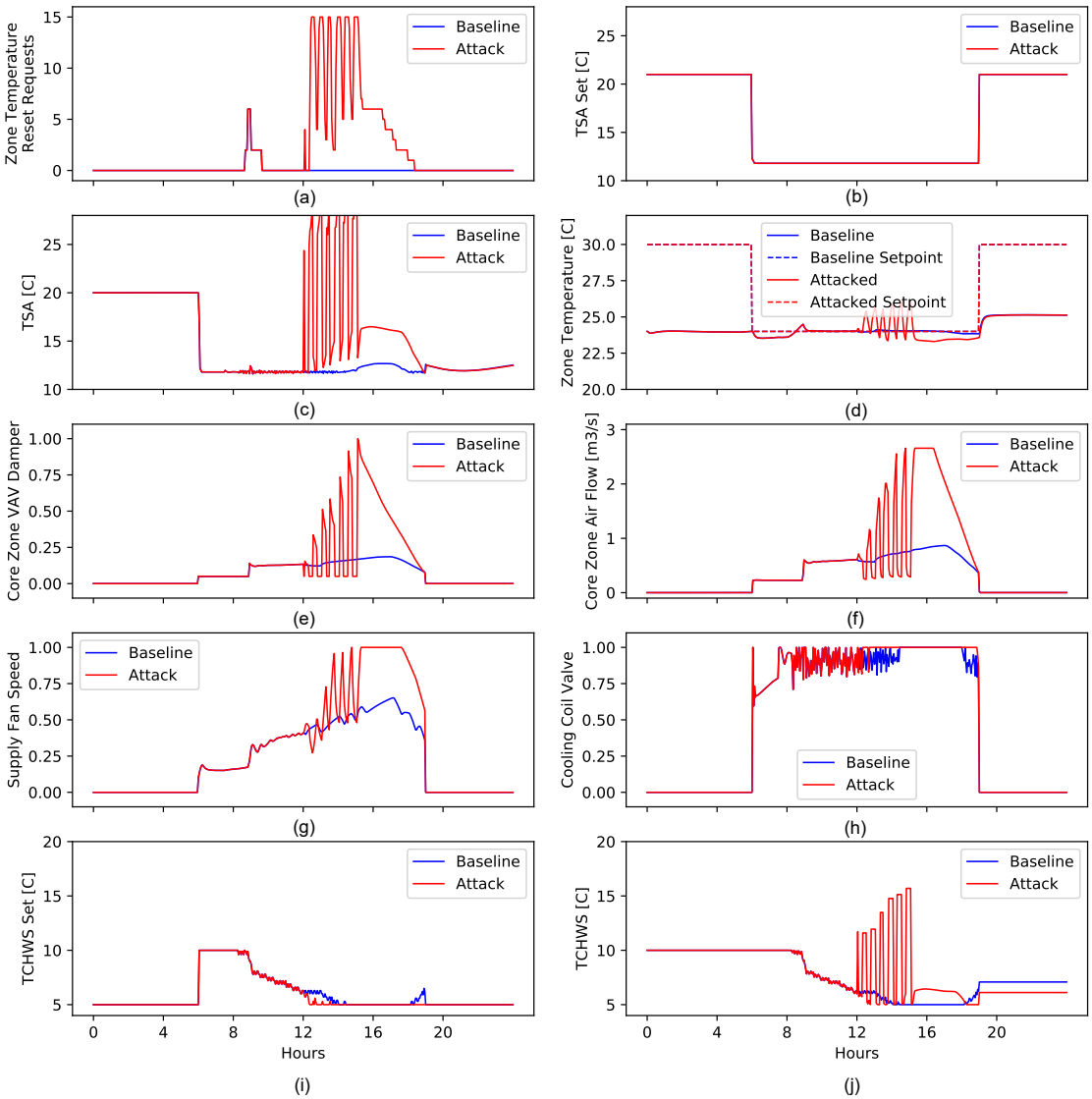

Figure 9: System states comparison between case 2 and baseline

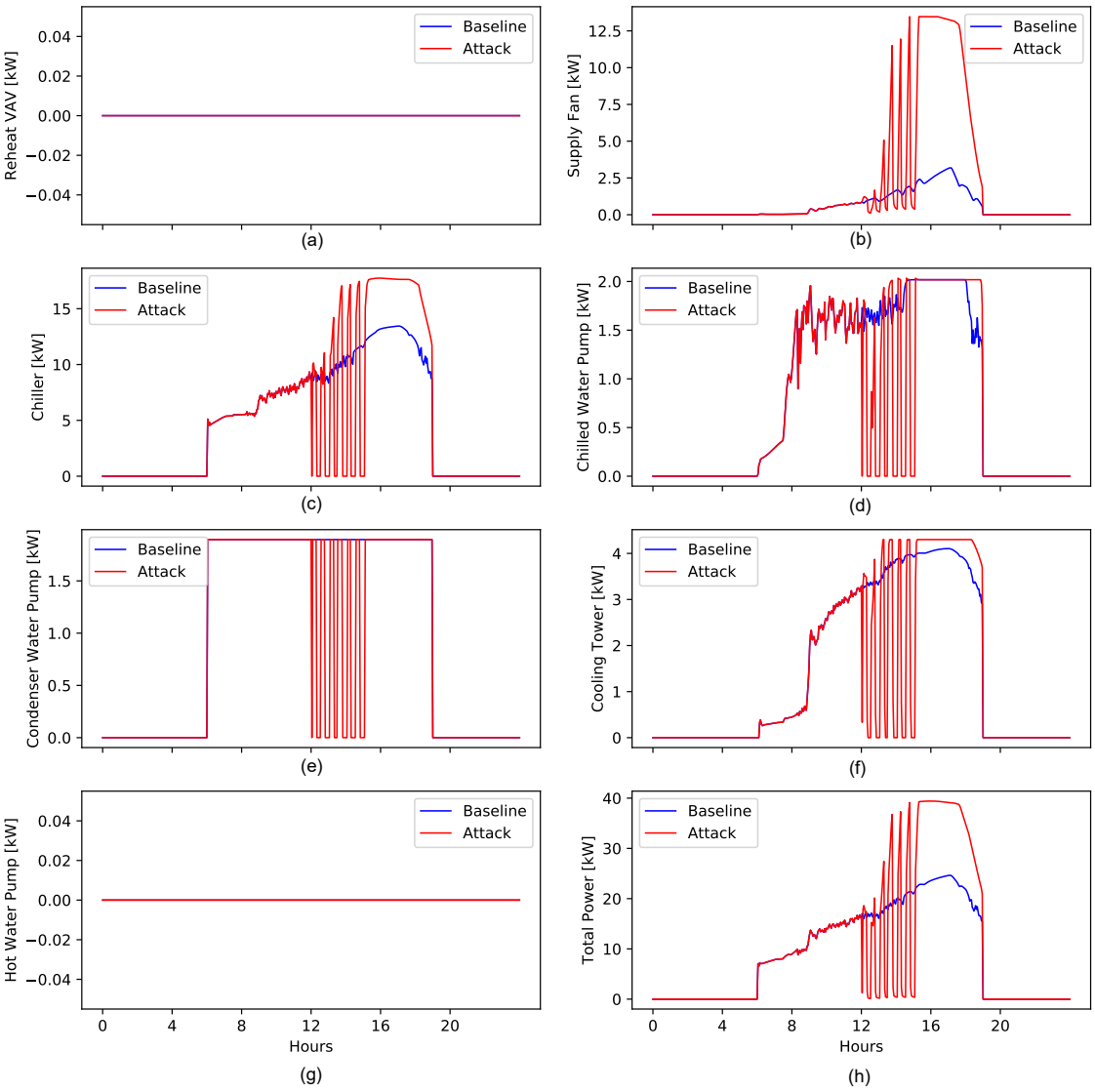

Figure 10: System power comparison between case 2 and baseline

3.3.3. Case 3

This scenario simulates a cyber-attack launched on the transmission of CHWST setpoint from the supervisory chilled water temperature controller to the local on-board chiller controller. During normal operations, the CHWST is continuously reset from a minimum value of 5 °C to a maximum value of 10 °C. The attack aims to block the chiller from receiving the setpoint. The attack is initiated at 10am and lasts for 3 hours. Detailed system responses are shown in Figure 11 and Figure 12.

During the attack period (10am-1pm), the baseline system gradually resets the CHWST from 7.5 °C at noon to around 5.5 °C in the afternoon because the cooling load in the building increases in the afternoon due to solar impact. The injected attack blocks the chiller from receiving the real-time CHWST setpoint starting from 10am. The chiller has to use a previously received setpoint that is 7.5 °C during the whole attack period. Therefore, the chiller only produces the chilled water at 7.5 °C even when the cooling load is large in the afternoon, and the cooling coil valve has to fully open in response to a high CHWST, which triggers the re-generation of CHWST in the reset controller (green dotted line in Figure 11(i)). However, the re-generated setpoint cannot reach the local chiller because of signal blocking. Furthermore, the high CHWST results in high SAT. To maintain the zone air temperature at the same setpoint of 24 °C, the VAV dampers open wider and the supply air fan increases its speed to deliver more air into the zones.

During the attack period, because of the higher CHWST, the chiller consumes less energy compared with the baseline system, which eventually results in less heat rejected in the cooling tower and thus less cooling tower fan power use. However, the chilled water pump consumes more energy than the baseline system because higher CHWST requires more pump energy to deliver more chilled water through the cooling coil to maintain the SAT setpoint. Moreover, the increased supply fan speed leads to increased fan power consumption.

3.3.4. Case 4

This scenario simulates a threat that lowers the global zone air temperature cooling setpoint during on-peak periods. The baseline cooling setpoint is 24 °C, and during the threatened period, the cooling setpoint is lowered to 22 °C. The threat is initiated at 1pm and lasts for 2 hours. Detailed system responses are shown in Figure 13 and Figure 14.

During the threat period, the zone air temperature setpoint for cooling is lowered from 24 °C to 22 °C, which increases the number of zone temperature reset requests as more cooling should be delivered to the zone, as defined in ASHRAE G36. However, although the number of the zone temperature reset requests increases, the SAT setpoint cannot be further reduced because of its lower limit. The lower zone temperature cooling setpoint introduces extra cooling needs to the chiller. However, the chiller controller has already reset its supply temperature setpoint to the lowest value (5.5 °C), the chiller cannot deliver chilled water with temperatures lower than the lowest limit to the AHU to meet the large cooling needs. Therefore, the SAT increases during the threat because of uncontrollable CHWST. Although the VAV terminal opens wider to introduce more air into the zone, the zone air temperature still cannot be lowered to the setpoint of 22 °C. The system eventually consumes more energy than the baseline during the threat due to the extra cooling needs handled by the chiller and its associated equipment (pumps, fan, cooling tower, etc.).

During the post-threat period, the zone temperature setpoint recovers from 22 °C to 24 °C, but the system states require more than 2 hours to recover to the baseline states. During the recovery, the system consume more energy than baseline due to the deviated states.

3.4. Multi-order Threat Injection

Simulating the above-mentioned threats simultaneously can help understand their combined impacts on the system. The power response of the system under a forth-order threat in Case 15 is shown in Figure 15. The power profile variations against the baseline power in Case 15 is almost a stack of additive variations of each single threat but additive effect is constrained by the capacity of the physical system. For example, during 6am-10am, because only Threat 1 is effective in Case 15, the power profile in Case 15 has the same response as that in Case 1. During 10am to 12pm, besides Threat 1, Threat 3 is also effective. The power deviations from the baseline in Case 15 is simply an additive stack of power deviations caused by Threat 1 and Threat 3 individually. During 12pm-1pm, Threat 1, 2, and 3 are effective simultaneously. The consequential influence on the power profile is close to an additive effect caused by all these three threats individually. During 1pm-3pm, Threat 1, 2, and 4 are activated. The resulting power profile in Case 15 has a weak additive effect compared with all threats individually, due to the fact that the system power cannot exceed its capacity limit.

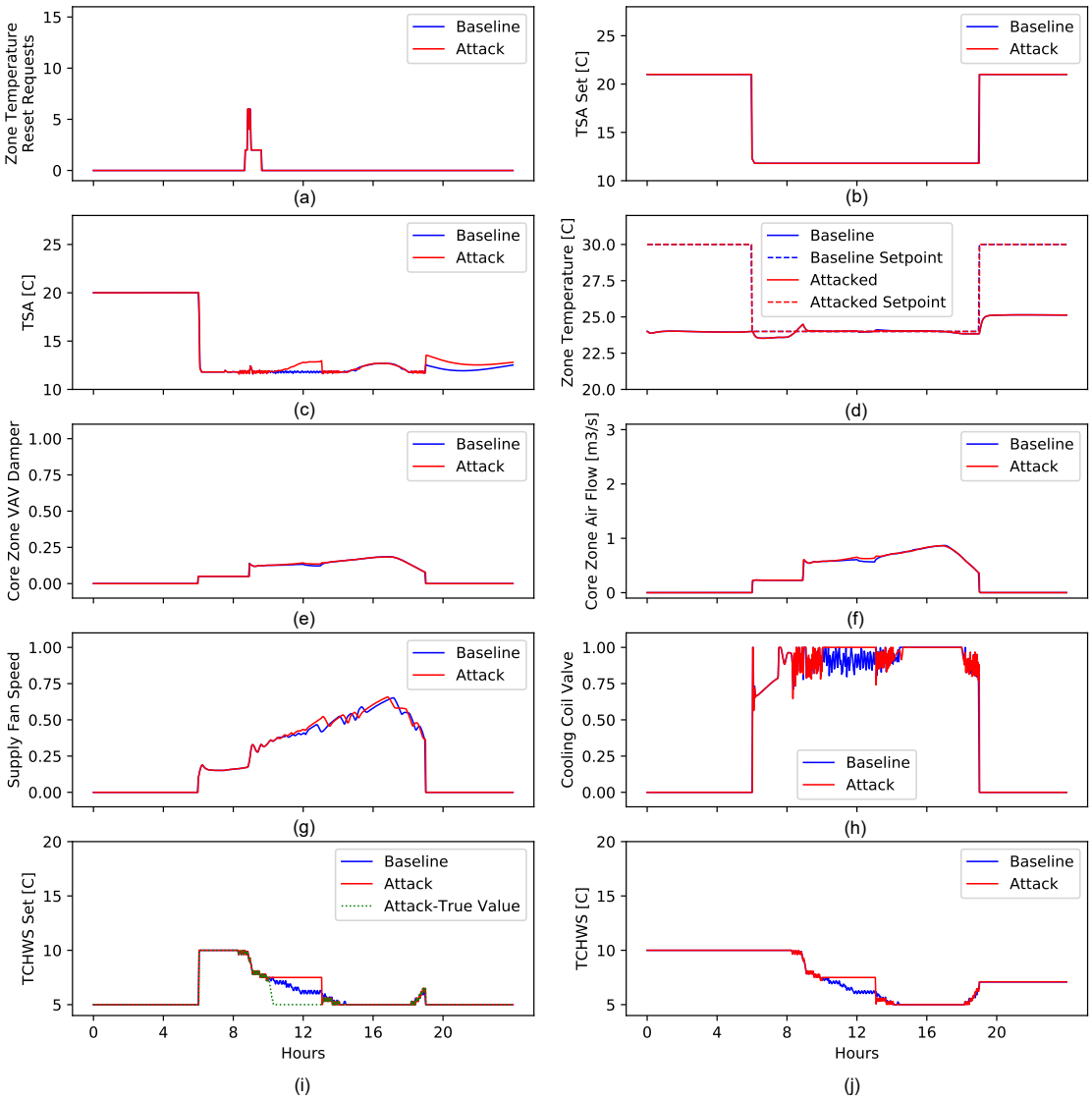

Figure 11: System states comparison between case 3 and baseline

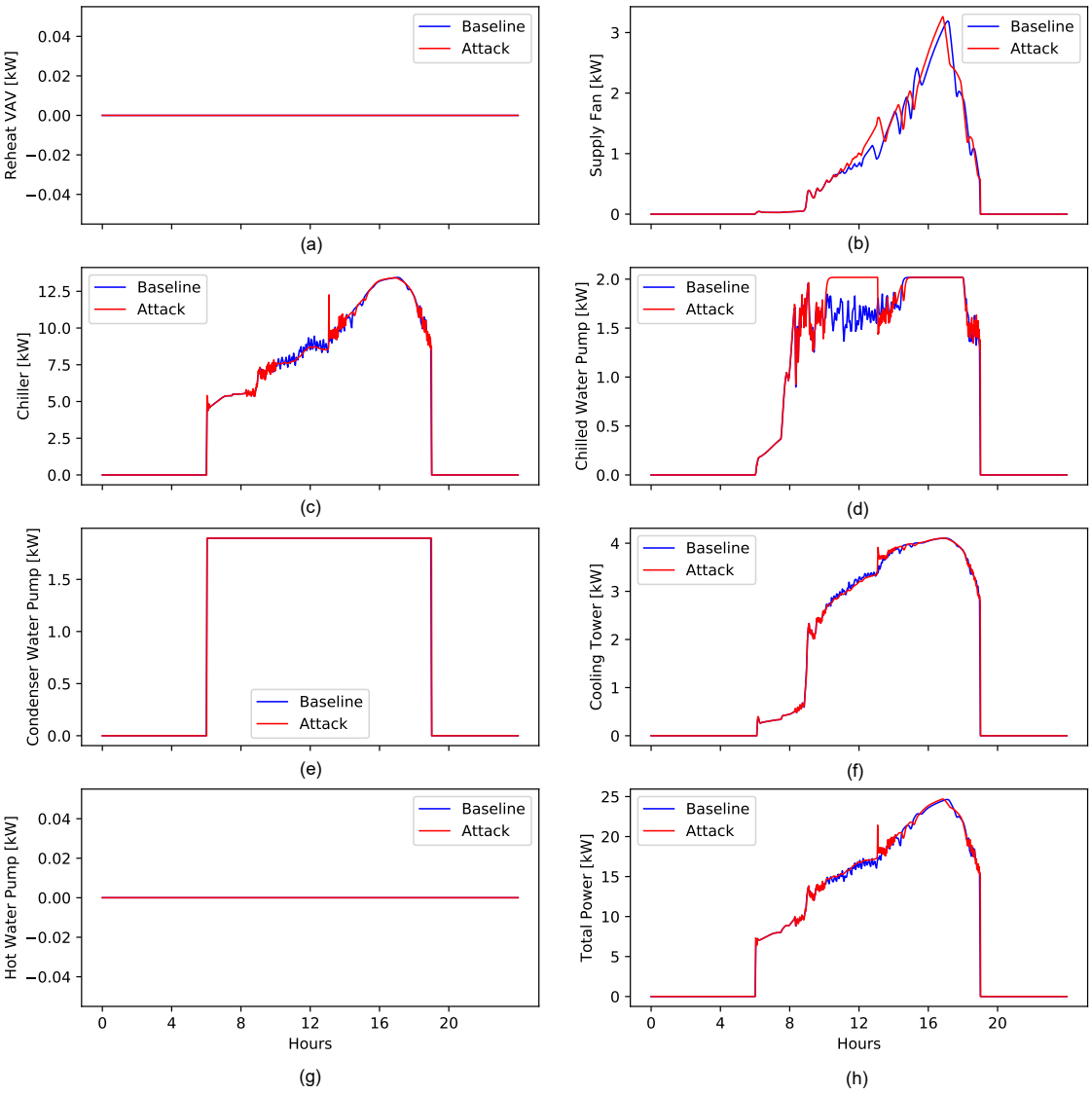

Figure 12: System power comparison between case 3 and baseline

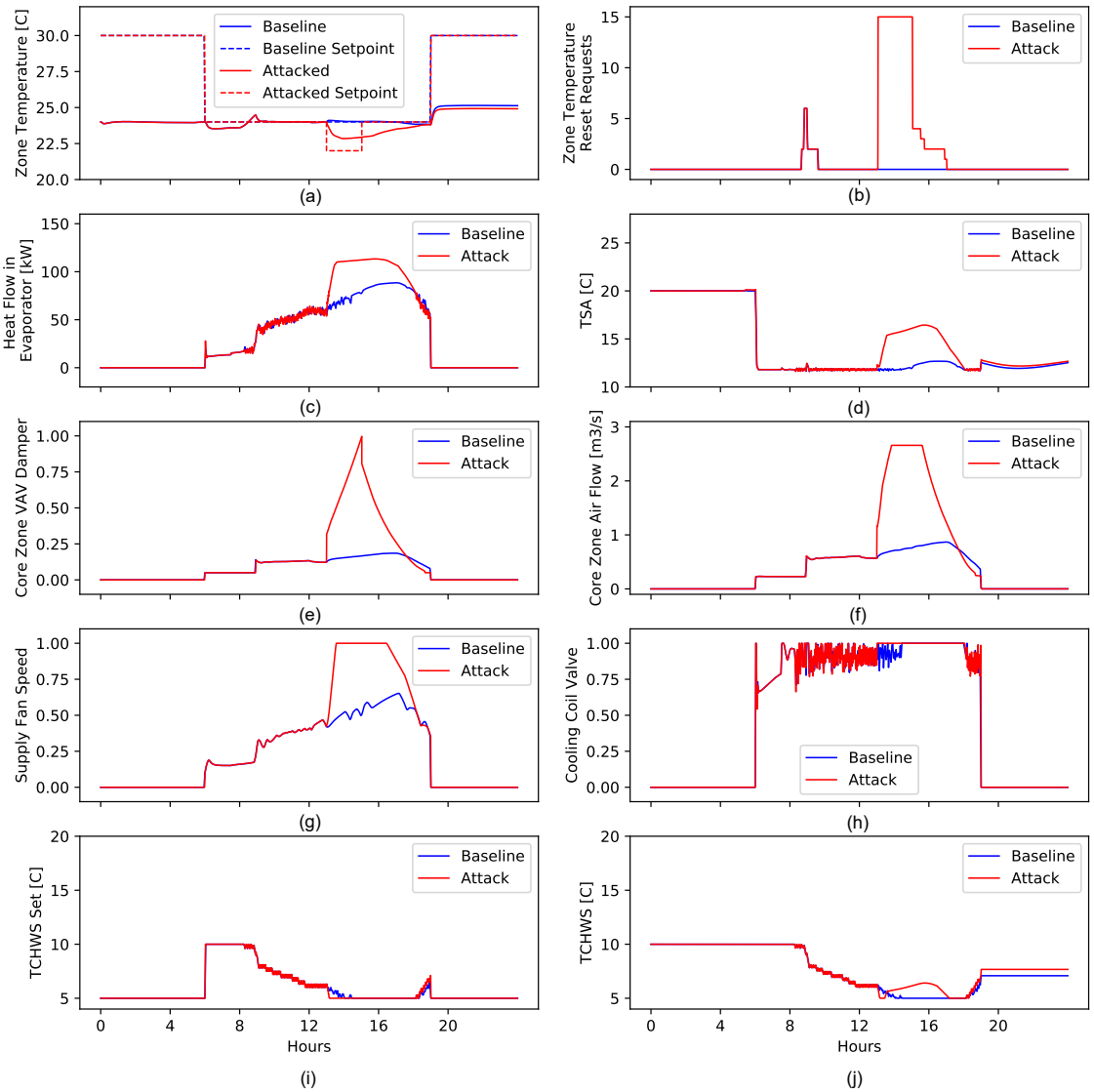

Figure 13: System states comparison between case 4 and baseline

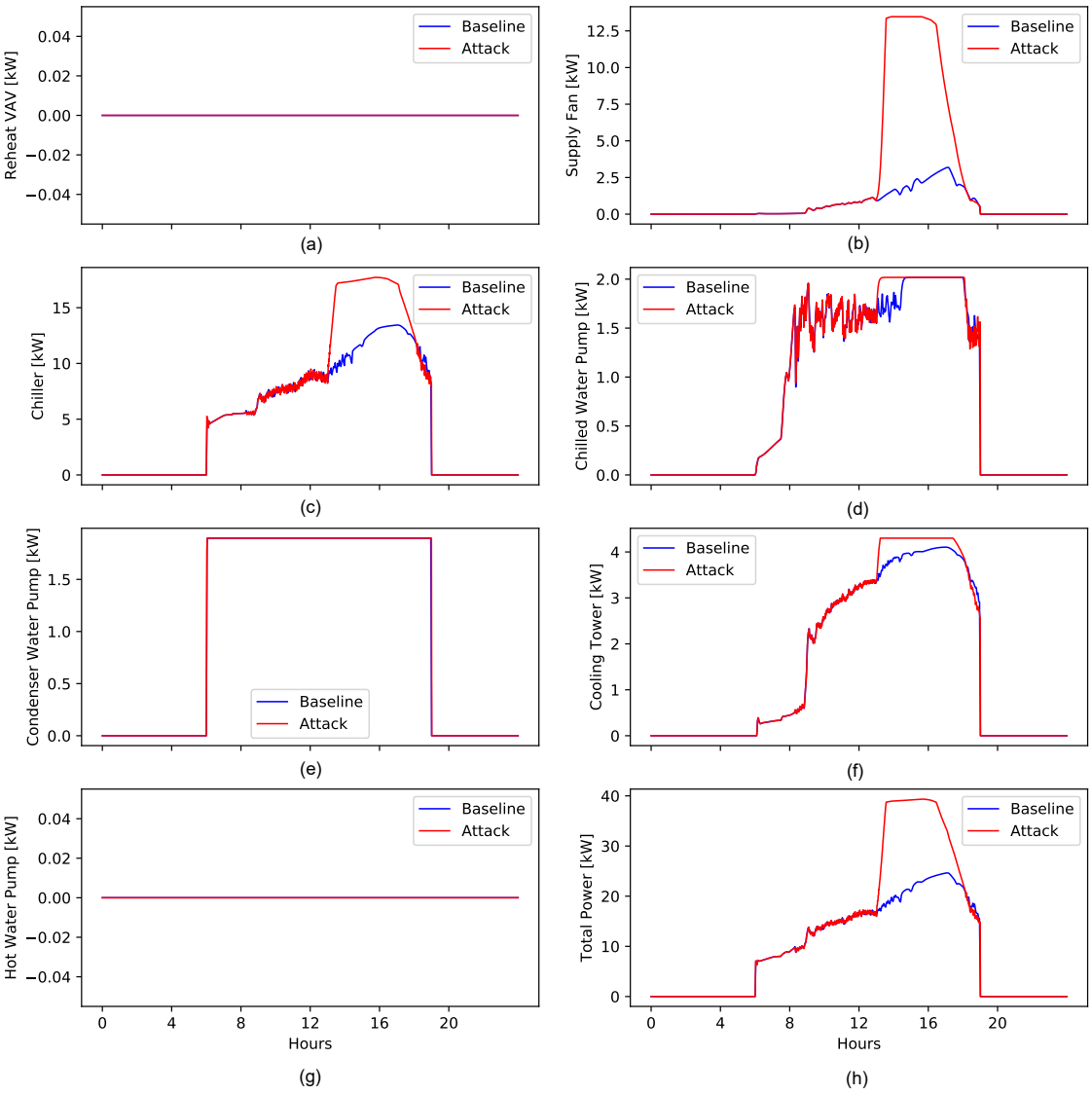

Figure 14: System power comparison between case 4 and baseline

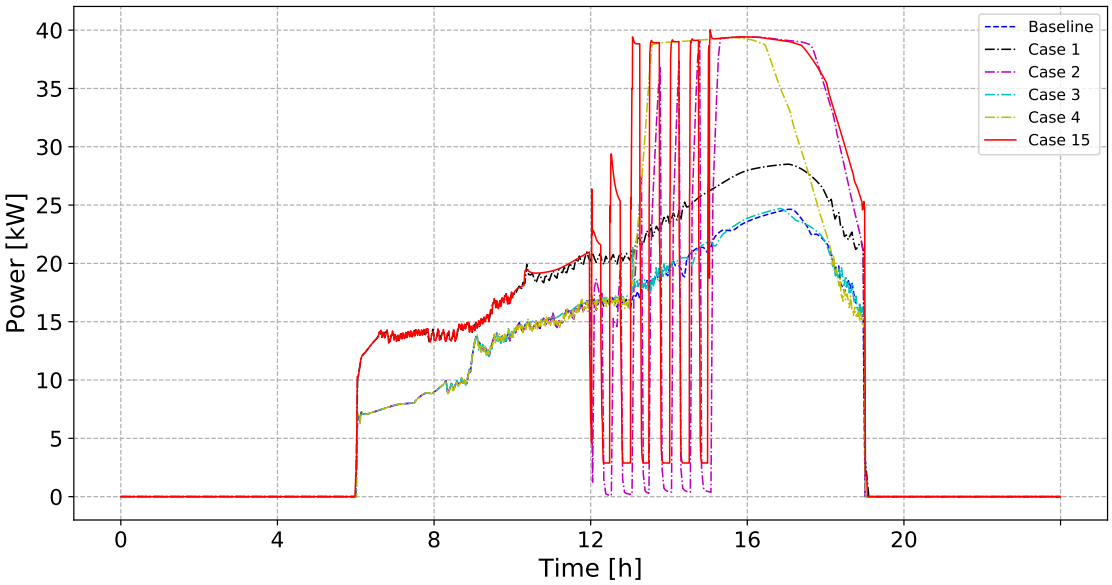

Figure 15: System power comparison among baseline, single-order threat and 4th-order threat

3.5. KPI Analysis

The KPIs for building service and grid service in each case are summarized and analyzed in this section. Building service metrics such as energy efficiency, power demand and total thermal discomfort are evaluated at the end of simulation period, and grid service metric such as demand flexibility is evaluated at an interval of one hour.

3.5.1. Building Service Quality

In terms of energy usage, Threat 4 in Case 4 has the most significant impact compared with other single threats. The two-hours of low zone air temperature setpoints led to 31.5% more energy use compared with the baseline. The extra energy usage mainly comes from the chiller and the supply air fan as shown in Figure 8. Threat 1 in Case 1 and Threat 2 in Case 2 increase the system energy use by 26.6% and 15.6% respectively. Threat 3 in Case 3 has the minimal energy influence on the system due to its small disturbances to the normal system operation. Multi-order threats have strong influences on the energy use as well but their impact is no stronger than the combined impact of each single threat. For example, in Case 13, the combined threats 1, 3 and 4 led to an energy use increase of 110 kWh (51.7%) compared with the baseline. Based on Table 3, the sum of the energy increase attributable to the individual threat 1, 3 and 4 is 338 kWh, is greater than that in Case 13 (298.3 kWh). This is due to the system constraints and the property of the injected threats. In particular, Threat 1 and Threat 4 have an overlapped influence on the supply air fan. Threat 1 operates the supply fan at a full speed during the occupied hours, and Threat 4 also leads to a full speed of the supply fan during the threatened period. Because the fan is already operating at its full capacity, no extra influence was observed. Furthermore, multi-order threats may have counteracting impact on each other. For example, Case 15 injected an additional Threat 2 compared with Case 13. Threat 2 alone in Case 2 resulted in an increase of 33.3 kWh of energy use. But its impact is significantly reduced when combined with the other threats in Case 15 compared with Case 13. The energy use is 23.5 kWh less in Case 15 than that of Case 13 even when one more threat is injected. The energy use increase in Case 2 with Threat 2 alone mainly comes from the post-threat period - a short period when the supply air fan works at its full capacity to recover the system states from deviations. Therefore, Threat 1 and Threat 2 have an overlapped impact of the fan energy use. And due to the short cycle of chiller operation, the system under Threat 2 consumes less energy during the threatened period, which contributes to the reduction of energy use in Case 15 in comparison with Case 13.

Table 3: Summary of impacts on energy efficiency, power demand and thermal discomfort

Metric	Baseline	Case														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Energy Efficiency(kWh)	212.6	268.8	245.9	214.5	279.7	289.6	270.9	320.6	247.9	251.6	282.8	293.9	322.5	298.3	253.6	299
Peak Demand (kW)	24.6	28.5	39.4	24.6	39.4	39.4	28.5	39.3	39.4	39.4	39.4	39.3	39.3	39.4	39.4	39.4
Total Discomfort (Kh)	0.0	0.0	7.0	0.0	0.2	6.3	0.0	0.2	7.0	7.1	0.2	6.4	6.4	0.3	7.1	6.5

For the peak power demand, Case 2 and Case 4 both result in a significant increase of 14.8 kW (60.2%) compared with the baseline, because in both cases, the system has to operate at its full capacity during the threat and/or the post-threat period. Case 1 also increases the demand by 15.9%, while Case 3 has no influence. In the cases where multi-order threats are injected, the maximum increase of demand is also 14.8 kW. No extra demand increase was observed because the system operates at its maximum capacity under threats.

For the total discomfort in all five zones, Threat 2 in Case 2 causes oscillating system states such as temperatures in both the water and the air loop of the HVAC system, which eventually leads to oscillating zone temperatures. In this case, the zone air temperatures deviate from their setpoints for almost 8.4 hours in total, which leads to a total discomfort of 7.0 Kh. Threat 4 alone causes 0.2 Kh of total discomfort during the threat and the post-threat period. Threat 1 and Threat 3 has insignificant influences on the total discomfort due to the "self-healing" function of the interactive control system. For the multi-order threats, the zone total discomfort is no greater than that of the sum of each single threat due to the overlapped impact.

3.5.2. Grid Service Quality

The summaries of hourly upward and downward demand flexibility for each case are listed in Table 4 and Table 5, respectively. Threat 1 in Case 1 alone has the minimal influence on the system demand flexibility.

Table 4: Summary of hourly upward demand flexibility in kW

Hour	Baseline	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	8.2	8.8	8.2	8.2	8.2	8.8	8.8	8.8	8.2	8.2	8.2	8.8	8.8	8.8	8.2	8.8
8	8.2	9.5	8.2	8.2	8.2	9.5	9.5	9.5	8.2	8.2	8.2	9.5	9.5	9.5	8.2	9.5
9	13.1	13.7	13.1	13.1	13.1	13.7	13.7	13.7	13.1	13.1	13.1	13.7	13.7	13.7	13.1	13.7
10	15.7	15.0	15.7	15.7	15.7	15.0	15.0	15.0	15.7	15.7	15.7	15.0	15.0	15.0	15.7	15.0
11	16.1	15.2	16.1	13.2	16.1	15.2	13.0	15.2	13.2	16.1	13.2	13.0	15.2	13.0	13.2	13.0
12	16.0	15.2	15.7	13.6	16.0	14.9	13.0	15.2	13.4	15.7	13.6	12.8	14.9	13.0	13.4	12.8
13	14.5	14.5	3.1	11.9	14.5	5.2	12.0	14.5	2.2	3.1	11.9	4.4	5.2	12.0	2.2	4.4
14	14.2	13.5	0.2	16.1	0.0	1.1	13.3	0.0	0.2	0.0	0.0	0.9	0.0	0.0	0.0	0.0
15	12.6	12.2	0.0	13.9	0.0	0.1	12.2	0.0	0.0	0.0	0.0	0.1	0.0	0.0	0.0	0.0
16	13.2	11.0	0.0	12.9	0.0	0.0	11.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	12.1	10.1	0.1	12.6	0.0	0.3	10.1	0.0	0.1	0.0	0.0	0.3	0.0	0.0	0.0	0.0
18	14.3	10.8	1.5	13.7	2.2	2.1	10.8	1.6	1.4	0.2	2.2	1.9	0.3	1.6	0.2	0.3
19	18.9	15.7	13.8	18.6	17.6	11.2	15.7	12.8	13.7	9.7	17.7	10.9	7.4	12.8	9.7	7.3
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Threat 2 in Case 2 influences the system demand flexibility during both the threat and the post-threat period. During the threat (12pm-3pm), the system can barely provide upward demand flexibility (the hourly value is almost 0 kW) because the chiller is remotely controlled to cycle on and off, and the system states mostly are out of control. For example, zone air temperatures during this threat oscillates above their setpoints due to the frequent activation and deactivation of the cooling source of the chiller. There is no way for the system to execute and fulfill an extra control command of providing upward flexibility by decreasing zone air temperature setpoint by 2 °C without available

chillers. The downward demand flexibility during this threat is decreased compared with the baseline. During the post-threat period (3pm-7pm), the cooling equipment firstly operate at their full capacities to recover the system states from the threat, and then gradually reduce their output as the states are being gradually recovered. Therefore, at the beginning of the post-threat (e.g., 3pm-5pm), the system cannot further increase its power output for the upward demand flexibility. When the states are gradually recovered, the ability of providing upward demand flexibility also is recovered. However, the system can provide more downward demand flexibility during the post-threat period due to the large power output for recovering the system states.

Table 5: Summary of hourly downward demand flexibility in kW

Hour	Baseline	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.3	0.0	0.3	0.3	0.3	0.0	0.0	0.0	0.3	0.3	0.3	0.0	0.0	0.0	0.3	0.0
9	3.6	0.7	3.6	3.6	3.6	0.7	0.7	0.7	3.6	3.6	3.6	0.7	0.7	0.7	3.6	0.7
10	4.8	3.2	4.8	4.8	4.8	3.2	3.2	3.2	4.8	4.8	4.8	3.2	3.2	3.2	4.8	3.2
11	5.8	4.9	5.8	7.5	5.8	4.9	6.1	4.9	7.5	5.8	7.5	6.1	4.9	6.1	7.5	6.1
12	6.1	5.4	5.9	7.6	6.1	5.3	6.6	5.4	7.4	5.9	7.6	6.5	5.3	6.6	7.4	6.5
13	8.1	7.6	5.1	9.2	8.1	6.3	8.5	7.6	4.7	5.1	9.2	5.7	6.3	8.5	4.7	5.7
14	9.3	8.7	5.0	9.5	0.0	7.4	8.9	0.0	5.3	0.0	0.1	7.2	0.0	0.1	0.0	0.1
15	10.2	10.3	2.6	10.6	0.0	3.4	10.3	0.0	2.5	0.0	0.0	3.1	0.0	0.0	0.0	0.0
16	11.5	11.3	10.2	11.5	0.0	9.0	11.3	0.0	10.1	0.0	0.0	8.7	0.0	0.0	0.0	0.0
17	12.4	11.7	15.3	12.5	14.2	12.5	11.7	10.8	15.2	11.7	14.2	12.3	8.8	10.8	11.7	8.8
18	11.5	11.3	18.5	11.6	18.5	14.1	11.3	14.5	18.5	16.9	18.5	14.0	12.9	14.6	16.8	12.9
19	8.0	6.9	10.8	8.1	8.3	9.3	6.9	8.6	10.8	13.8	8.2	9.4	11.6	8.6	13.8	11.6
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Threat 3 in Case 3 blocks the chiller from receiving CHWST setpoint and a previous setpoint of 7.5 °C is used during the threat period (10am to 1pm). During the threat, When asked for providing upward flexibility by decreasing zone temperature setpoint by 2 °C, the system needs to provide more cooling by lowering CHWST setpoint. But due to the blocking of CHWST setpoints, the chiller cannot be controlled to increase power outputs for harvesting upward demand flexibility. Therefore, compared with the baseline system, this threat-injected system provides about 3 kW less upward flexibility. However, this threat can slightly increase the system's capability of providing downward demand flexibility by about 1.5 kW per hour. Because HAVC equipment other than the chiller operate at a higher capacity in the threatened system than that in the baseline system, they can reduce more outputs when downward demand flexibility is called. As for the post-threat period, this threat has no significant influence due to the availability of all HVAC equipment.

Threat 4 in Case 4 assumes a temporary loss of control of zone temperature setpoint by the BAS from 1pm to 3pm, which leads to the unavailability of resetting zone temperature setpoint for harvesting both upward and downward demand flexibility. Therefore, during the threat, the system can provide no flexibility. At the beginning of the post-threat period (roughly 3pm - 4pm), the system regains the control of zone temperature setpoint, and works at its maximum capacity to recover from deviated states. Upward flexibility cannot be provided because the system cannot further increase its power output. Downward flexibility cannot be provided because the system is already working at its best to increase zone temperature for recovery. As the system gradually gets recovered (from 4pm-7pm), the ability of providing upward and downward flexibility gets recovered as well. Due to the deviated system states, the system can provide more downward but less upward demand flexibility than the baseline.

For multi-order threats, their impacts on the demand flexibility is less than the additive effects of combining single threats. The reason is a system has a minimum and maximum power profile that it should operate within due to control needs and system capacity. No matter how many threats are injected, the possible minimum and maximum profiles will not be affected. For example, Threat 4 alone leads to the unavailability of providing neither upward nor

downward flexibility. When the system is injected with Threat 4 and other threats, the maximum adverse effect would still be the total loss of ability of providing flexibility as shown in Case 7, Case 9 and Case 12 ~ Case 15.

4. Conclusions

This paper presents a scalable and generic threat injection framework for grid-interactive efficient buildings (GEBs). The framework is demonstrated on a Modelica-based energy simulator. A numerical case study is conducted to demonstrate the capability of the framework for supporting single- and multi-order threat modeling and simulation. Four threats and their various combinations are injected at the on-peak period but have different start time and periods. The GEB's responses under threats are quantified by different key performance indexes, which are also summarized and analyzed. Simulation results show that the threat that operates maliciously the supply fan at the full speed all time results in 26.6% more energy use and 15.9% more peak demand, but has small influence on the upward and downward demand flexibility. The threat that remotely forces the chiller to cycle on and off during peak hours results in 15.6% more energy use, 60% more peak demand and an significant increase of thermal discomfort. The system under this threat cannot provide upward demand flexibility and the downward demand flexibility is halved. The cyber-attack that leads to short-term signal blocking has a small influence on the system operation due to the resilience of the HVAC interactive control system. The threat that lowers the global zone temperature setpoint can lead to 31.5% more energy use, 60% more power demand, and an significant increase of zone total thermal discomfort. The combination of the above four threats have combinative effects on the system but the effects are no stronger than the additive effects of all single threat.

Future work includes to extend this framework for other FMU-supported energy simulators such as EnergyPlus. For EnergyPlus, the required modifications are to create a threat-free EnergyPlus template. To construct a similar three-free template in EnergyPlus, the EnergyPlus EMS will be utilized to define a similar *Overwrite* package as in Modelica but with EMS-specific syntax. For example, when an EnergyPlus variable is to be overwritten for a threat injection, EMS is used to define the variable name. To generate an EnergyPlus/FMU wrapper with overwritten variables, the EnergyPlus syntax will be coded in the parser to automatically detect the overwritten variables as defined in the EMS, and create a wrapper using the EMS. The wrapper, similarly as in Modelica, will have two parts: one is the overwritten flag indicating if the variable should be overwritten at the current time step, and the other is the corrupted value for the overwritten variable. When the overwritten flag is activated, EnergyPlus will receive external signals through the EMS. Otherwise, EnergyPlus will receive 0 and keep using its original signals. With these modifications, an EnergyPlus-specific threat-free FMU can then be used with the proposed threat injection framework.

5. Acknowledgement

The research reported in this paper was supported by the Building Technologies Office at the U.S. Department of Energy through the Emerging Technologies program under award number DE-EE0009150.

References

[1] U. DOE, Chapter 5: Increasing efficiency of building systems and technologies, Quadrennial Technology Review: An Assessment of Energy Technologies and Research Opportunities (2015) 143–181.

[2] M. Neukomm, V. Nubbe, R. Fares, Grid-interactive efficient buildings technical report series: Overview of research challenges and gaps, Report, National Renewable Energy Lab.(NREL), Golden, CO (United States) (2019).

[3] M. R. Brambley, S. Katipamula, Commercial building retuning: a low-cost way to improve energy performance, ASHRAE Journal 51 (10) (2009) 12–20.

[4] J. G. Allen, P. MacNaughton, U. Satish, S. Santanam, J. Vallarino, J. D. Spengler, Associations of cognitive function scores with carbon dioxide, ventilation, and volatile organic compound exposures in office workers: a controlled exposure study of green and conventional office environments, Environmental health perspectives 124 (6) (2016) 805–812.

[5] M. Peacock, Anomaly detection in bacnet/ip managed building automation systems, Ph.D. thesis, Edith Cowan University (2019).

[6] M. Honorof, Building hack almost landed google in hot water (5 2013).
URL `https://www.nbcnews.com/id/wbna51805522`

[7] J. Vijayan, Target attack shows danger of remotely accessible hvac systems (2 2014).
URL `https://www.computerworld.com/article/2487452/target-attack-shows-danger-of-remotely-accessible-hvac-systems.html`

[8] A. Roth, J. Reyna, Grid-interactive efficient buildings technical report series: Whole-building controls, sensors, modeling, and analytics, Report, National Renewable Energy Lab.(NREL), Golden, CO (United States) (2019).
[9] Y. Li, Z. O'Neill, A critical review of fault modeling of hvac systems in buildings, Building Simulation 11 (5) (2018) 953–975.
[10] S. Li, J. Wen, Development and validation of a dynamic air handling unit model, part i, ASHRAE Transactions 116 (1) (2010).
[11] S. Li, J. Wen, X. Zhou, C. J. Klaassen, Development and validation of a dynamic air handling unit model, part 2, ASHRAE Transactions 116 (1) (2010) 57–73.
[12] H. Cheung, J. E. Braun, Empirical modeling of the impacts of faults on water-cooled chiller power consumption for use in building simulation programs, Applied Thermal Engineering 99 (2016) 756–764.
[13] H. Cheung, J. E. Braun, Development of fault models for hybrid fault detection and diagnostics algorithm: October 1, 2014–may 5, 2015, Report, National Renewable Energy Lab.(NREL), Golden, CO (United States) (2015).
[14] M. Basarkar, Modeling and simulation of hvac faults in energyplus (2011).
[15] R. Zhang, T. Hong, Modeling of hvac operational faults in building performance simulation, Applied Energy 202 (2017) 178–188.
[16] Y. Li, Z. O'Neill, An innovative fault impact analysis framework for enhancing building operations, Energy and Buildings 199 (2019) 311–331.
[17] R. Khire, M. Trcka, Model based failure mode effect analysis on whole building energy performance, in: 13th Conference of International Building Performance Simulation Association, 2013.
[18] Y. Chen, S. Huang, D. Vrabie, A simulation based approach to impact assessment of physical faults: large commercial building hvac case study, in: 2018 Building performance modeling conference and SimBuild co-organized by ASHRAE and IBPSA-USA. Chicago, IL, USA, 2018.
[19] D. G. Holmberg, D. Evans, BACnet wide area network security threat assessment, US Department of Commerce, National Institute of Standards and Technology, 2003.
[20] J. Kaur, J. Tonejc, S. Wendzel, M. Meier, Securing bacnet's pitfalls, in: IFIP International Information Security and Privacy Conference, Springer, 2015, pp. 616–629.
[21] M. N. Johnstone, M. Peacock, J. den Hartog, Timing attack detection on bacnet via a machine learning approach, in: 13th Australian Information Security Management Conference, 2015, pp. 57–64.
[22] J. Tonejc, S. Güttes, A. Kobekova, J. Kaur, Machine learning methods for anomaly detection in bacnet networks, J. UCS 22 (9) (2016) 1203–1224.
[23] B. Bowers, How to own a building: Exploiting the physical world with bacnet and the bacnet attack framework, ShmooCon (2013).
[24] A. Householder, A. Manion, L. Pesante, G. M. Weaver, R. Thomas, Managing the threat of denial-of-service attacks, Report, CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST (2001).
[25] Y.-L. Huang, A. A. Cárdenas, S. Amin, Z.-S. Lin, H.-Y. Tsai, S. Sastry, Understanding the physical and economic consequences of attacks on control systems, International Journal of Critical Infrastructure Protection 2 (3) (2009) 73–83.
[26] S. Sridhar, M. Govindarasu, Model-based attack detection and mitigation for automatic generation control, IEEE Transactions on Smart Grid 5 (2) (2014) 580–591.
[27] X. Lou, C. Tran, R. Tan, D. K. Yau, Z. T. Kalbarczyk, Assessing and mitigating impact of time delay attack: a case study for power grid frequency control, in: Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems, 2019, pp. 207–216.
[28] S. Sridhar, G. Manimaran, Data integrity attacks and their impacts on scada control system, in: IEEE PES general meeting, IEEE, 2010, pp. 1–6.
[29] A. Sheikh, V. Kamuni, A. Patil, S. Wagh, N. Singh, Cyber attack and fault identification of hvac system in building management systems, in: 2019 9th International Conference on Power and Energy Systems (ICPES), 2019, pp. 1–6. doi:10.1109/ICPES47639.2019.9105438.
[30] M. Wetter, W. Zuo, T. S. Nouidui, X. Pang, Modelica buildings library, Journal of Building Performance Simulation 7 (4) (2014) 253–270.
[31] T. Nouidui, M. Wetter, W. Zuo, Functional mock-up unit for co-simulation import in energyplus, Journal of Building Performance Simulation 7 (3) (2014) 192–202.
[32] DOE, Python ems: A unique energyplus feature gets a serious upgrade (7 2020).
URL https://www.energy.gov/eere/buildings/articles/python-ems-unique-energyplus-feature-gets-serious-upgrade
[33] P. Fritzson, Principles of object-oriented modeling and simulation with Modelica 3.3: a cyber-physical approach, John Wiley & Sons, 2014.
[34] Y. Fu, M. Wetter, W. Zuo, Modelica models for data center cooling systems, in: 2018 Building Performance Analysis Conference and SimBuild, Chicago, Illinois, United States of America, 2018.
[35] Y. Fu, W. Zuo, M. Wetter, J. W. VanGilder, X. Han, D. Plamondon, Equation-based object-oriented modeling and simulation for data center cooling: A case study, Energy and Buildings 186 (2019) 108–125.
[36] Y. Fu, W. Zuo, M. Wetter, J. W. VanGilder, P. Yang, Equation-based object-oriented modeling and simulation of data center cooling systems, Energy and Buildings 198 (2019) 503–519.
[37] Y. Fu, X. Lu, W. Zuo, Modelica models for the control evaluations of chilled water system with waterside economizer, in: Proceedings of the 13th International Modelica Conference, Regensburg, Germany, March 4–6, 2019, no. 157, Linköping University Electronic Press, 2019.
[38] Y. Fu, X. Han, K. Baker, W. Zuo, Assessments of data centers for provision of frequency regulation, Applied Energy 277 (2020) 115621.
[39] L. Helsen, D. Blum, F. Jorissen, S. Huang, Y. Chen, J. Arroyo, K. Benne, Y. Li, V. Gavan, L. Rivalin, et al., Prototyping the boptest framework for simulation-based testing of advanced control strategies in buildings, in: BS2019, Date: 2019/09/02-2019/09/05, Location: Rome, 2019.
[40] M. Wetter, U. DOE, Buildingspy (4 2019). doi:10.11578/dc.20190430.2.
URL https://www.osti.gov//servlets/purl/1569219
[41] J. Åkesson, M. Gäfvert, H. Tummescheit, Jmodelica—an open source platform for optimization of modelica models, in: Proceedings of MATHMOD, 2009.
[42] A. K. Athienitis, N. Morovat, J. Date, Development of a dynamic energy flexibility index for buildings and their interaction with smart grids, in: 2020 Summer Study on Energy Efficiency in Buildings, 2020.
[43] U. Department of Energy, Commercial prototype building models (4 2021).
URL https://www.energycodes.gov/development/commercial/prototype_models

[44] ASHRAE, ASHRAE Guideline 36P, High Performance Sequences of Operation for HVAC systems, First Public Review Draf, ASHRAE, 2016.
[45] M. Wetter, J. Hu, M. Grahovac, B. Eubanks, P. Haves, Openbuildingcontrol: Modeling feedback control as a step towards formal design, specification, deployment and verification of building control sequences, in: Building Performance Modeling Conference and SimBuild, 2018.
[46] S. T. Taylor, B. Gill, R. Kiriu, Ashrae research project 1711: Advanced sequences of operation for hvac systems – phase ii central plants and hydronic systems task 5: Reporting of findings (2019).