Attack Graph Implementation and Visualization for Cyber Physical Systems

Authors:

Mariam Ibrahim, Qays Al-Hindawi, Ruba Elhafiz, Ahmad Alsheikh, Omar Alquq

Date Submitted: 2020-02-02

Keywords: vehicular networks, industrial control system, attack graph, cyber physical system security

Abstract:

Cyber-attacks threaten the safety of cyber physical systems (CPSs) as a result of the existence of weaknesses in the multiple structural units constituting them. In this paper, three cyber physical systems case studies of a pressurized water nuclear power plant (NPP), an industrial control system (ICS), and a vehicular network system (VNS) are examined, formally presented, and implemented utilizing Architecture Analysis and Design Language, determining system design, links, weaknesses, resources, potential attack instances, and their pre-and post-conditions. Then, the developed plant models are checked with a security property using JKind model checker embedded software. The attack graphs causing plants disruptions for the three applications are graphically visualized using a new graphical user interface (GUI) windows application.

processes

MDPI

Article

Attack Graph Implementation and Visualization for Cyber Physical Systems †

Mariam Ibrahim [1,*], Qays Al-Hindawi [1,2], Ruba Elhafiz [1], Ahmad Alsheikh [1,3] and Omar Alquq [1]

1 Department of Mechatronics Eng., German Jordanian University, Amman 11180, Jordan; qays.al-hindawi@uni-wuppertal.de (Q.A.-H.); r.elhafiz@gju.edu.jo (R.E.); a.alsheikh@gju.edu.jo (A.A.); o.alquq@gju.edu.jo (O.A.)
2 School of Electrical, Information and Media Eng., University of Wuppertal, 42119 Wuppertal, Germany
3 Department of Mechanical Eng., and Mechatronics, Deggendorf Institute of Technology, 94469 Deggendorf, Germany
* Correspondence: Mariam.wajdi@gju.edu.jo

† This paper is an extended version of paper published in the international conference: 2018 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, 28–30 June 2018.

Received: 14 November 2019; Accepted: 17 December 2019; Published: 20 December 2019

Abstract: Cyber-attacks threaten the safety of cyber physical systems (CPSs) as a result of the existence of weaknesses in the multiple structural units constituting them. In this paper, three cyber physical systems case studies of a pressurized water nuclear power plant (NPP), an industrial control system (ICS), and a vehicular network system (VNS) are examined, formally presented, and implemented utilizing Architecture Analysis and Design Language, determining system design, links, weaknesses, resources, potential attack instances, and their pre-and post-conditions. Then, the developed plant models are checked with a security property using JKind model checker embedded software. The attack graphs causing plants disruptions for the three applications are graphically visualized using a new graphical user interface (GUI) windows application.

Keywords: cyber physical system security; attack graph; industrial control system; vehicular networks

1. Introduction

Modeling cyber-attack is an important issue for securing cyber physical systems (CPSs). The pervasive smart environment in Internet of Things (IoTs) has the potential to monitor various human activities by using smart devices. For instance, [1] presented an approach for detecting sleep attacks due to immune system attacks, thus affecting daily activities measured using the S-band sensing method. The integration of wireless body area networks (WBANs) with recent cloud and sensor technologies offers significant improvement in the efficiency and the functionality of medical and health care systems [2]. This integration can preserve the cost of medical services and allow a wide distribution of medical knowledge to nonmedical personnel [3]. However, deploying IoT devices in organizations can greatly affect their security, consequently bringing down or disrupting their business operations.

A brief overview is presented by [4] on state-of-the-art research trends in the area of IoT operating system (OS) management: opportunities, challenges, and solutions. This includes IoT energy efficiency, real time capabilities, network connectivity, security and safety, small memory footprint, heterogeneous devices support, intelligent IoT, and IoT and big data.

Attack graphs are conceptual diagrams used to analyze how a target can be attacked. The main advantage of an attack graph is that it helps to identify any possible attacks on the system [5], hence aiding designers in implementing prevention strategies through making risk analysis [6]. The novelty of this work lies in presenting a model-based technique for implementing attack graphs for three cyber physical systems (CPSs): a nuclear power plant (NPP) control system, an industrial control system (ICS), and a vehicular network system (VNS). This demands a general specification of systems models and the security properties being studied. The models and their security properties are encoded using Architecture Analysis and Design Language (AADL) [7] and verified using JKind model checker embedded software [8]. The generated graphs are visualized using a new graphical user interface (GUI) windows application implemented with C# language in Microsoft Visual Studio (VS) [9].

This paper extends the conference version [10] significantly by introducing a newly developed visualizer Windows application that creates a GUI that is encoded using C# within Microsoft VS. The JKind model checker generates significantly large spreadsheet attack scenarios. Therefore, the visualizer Windows application manages the number of rows of the produced attack scenarios and represents them in an appealing, user friendly, graphical way. Simulation results are shown for the NPP system, the ICS, and the VNS.

The contribution of this work is twofold. First, system models are formalized for three CPS case studies. The formal descriptions capture the architectural specifications of the systems—their elements and their connectivity. In addition to that, the descriptions capture systems behaviors—their dynamic state variables, pre and post conditions of the attacks instances, and the security properties. The models are encoded using AADL, translated to Lustre using Assume Guarantee Reasoning Environment (AGREE), and finally checked using JKind. The generated attack scenarios are fed to a newly developed visualizer Windows application that can graphically illustrate a specific attack sequence of interest, its spread sheet, and the complete attack graph. The remainder of this paper is assembled as follows. Section 1.1 revises the related work. Section 2 describes the NPP system. Section 3 presents attack graph implementation and visualization against the NPP. Section 4 represents the implementation of cyber-attack scenarios against the ICS. Section 5 represents the implementation of cyber-attack scenarios against the VNS. Section 6 summarizes and presents some future directions.

1.1. Related Work

Various papers have been explored in the literature on attack graph modeling and analysis for CPS. For instance, [11] proposes a method on security standards representation in the shape of a graph. The method also introduces minimum-security baseline extraction with currently implemented controls by means of a vertex cover algorithm.

A reachability hyper-graph partitioning is performed by [12] to direct the distributed search to implement an attack graph for a network. The tasks allocated to each agent are defined by conducting partitioning to the reachability hyper-graph. Each distributed agent must apply attacker privilege identification tasks for a number of closely connected networked software applications.

A probabilistic threat modeling approach, pwnPr3d, is proposed by [13] for automatic attack graph generation based on network modeling. PwnPr3d automatically generates probability distributions over the time to compromise (TTC) for each asset in the system. In [14], a taxonomy of the implemented algorithms and models related with a reachability study, attack graph modeling, and core building phases of attack graph is developed. The proposed taxonomy takes into consideration the utilization of attack graphs for network security.

An adapted attack graph (AAG) is compared by [15] with the fault tree approach to identify which of the two methods is more effective at helping cyber-attack insight. While the two methods have some conceptual commonalities, they have different symbol constructions and data flow. The paper found that the AAG approach is more efficient than the fault tree approach at aiding cyber-attack perception.

An attack-graph-based analysis is presented by [16] of threats on patient controlled analgesia (PCA) interoperability. Considering a trusted coordinator, most of the attacks are discovered to be various shapes of the confused deputy attack. The analysis shows that individual medical device safety does not equate to interoperable medical devices (IMDs) safety, despite having a trusted coordinator.

A method to model components of the cyber-physical architecture of the vehicle is proposed by [17] using graphs. The model captures the security policy employed as well as vulnerability information and access rights. An attacker model is considered as a set of attacks originating from all the attack vectors (short range, long range, and indirect physical access). The system and the attacker are modeled with behavioral rules using a graph transformation system.

A novel approach is introduced by [18] for alert correlation based on graphs and absorbing Markov chains. The approach answers most of the challenges that a correlation system is faced with; the context-based management system ensures the portability of the system and reduces false positive alerts, and the correlation system guarantees real-time and scalability properties.

Attack graph modeling is demonstrated by [19] on a theoretical ambulatory medical device. The paper highlights the need to model ambulatory devices by demonstrating specific attack vectors that show greater risk to ambulatory devices.

An extended network modeling is introduced by [20] for the Multi-host, Multi-stage Vulnerability Analysis (MulVAL) framework. The proposed modeling considers the physical network topology, the supported short-range communication protocols, and the modeled specific industrial communication architectures. Furthermore, various network attacks are modeled, including bus spoofing, wired equivalent privacy (WEP) cracking, Bluetooth personal identification number (PIN) cracking, address resolution protocol (ARP) spoofing, and domain name server (DNS) spoofing.

HERCULE, an automated multi-stage intrusion analysis system, is presented by [21] to reconstruct a complete and understandable attack story from multiple correlated logs. HERCULE automatically generates a multi-dimensional weighted graph with valuable information grouped within. The proposed graph provides a "panoramic view" of the logs implemented by multiple system components.

The Safelite framework constructed by [22] semi automatically converts an attack graph to a hierarchical attack representation model (HARM). A HARM is generated from the vulnerability scan reports to eliminate the state-space explosion problem. The paper incorporates the attack modeling and the security risk evaluation using the HARM.

Two modified Bayesian networks (BN)-based attack graph models are utilized by [23] to determine the probabilities of successful attacks on the power system. The models consider different cyber-attack paths, skill levels of attackers, and mean time to compromise (MTTC) of successful attacks. It is found that, as more known vulnerabilities are exploited, smaller MTTC results.

A comprehensive risk management technique is provided by [24] to a smart grid CPS. The example shows that the technique sufficiently allows the organization to analyze their security issues, identify critical assets, and assess vulnerabilities and potential threats. A method is proposed by [25] for determining the risk of IoT device deployment using an augmented attack graph. The results show the potential risk in using IoT devices in organizations and illustrate that randomly employing devices can greatly affect the security of the organization's network. The heuristic approach suggests that the possible risk of two deployed devices is greater than or equal to the sum of their individual risk scores.

A framework is proposed by [26] for security modeling and assessment of the IoT. The framework is used to implement a graphical security model to automate the security analysis of an IoT. The framework encompasses five steps: preprocessing, security model generation, visualization and storage, security analysis, and changes and updates. In this framework, an IoT generator, a security model generator, and a security evaluator are developed.

A model and a methodology are presented by [27] for security risk analysis of enterprise networks using probabilistic attack graphs. This model annotates the attack graph with known vulnerabilities and their likelihoods of exploitation. By disseminating the exploit likelihoods through the attack graph, a metric is evaluated that determines the security risk of enterprise networks.

A suite of metrics is described by [28] for determining overall network security risk based on overall attack graph. These metrics are gathered into families, which are combined into a risk metric for the network. A simulation-driven approach is developed by [29] for secure information system design. This method can be utilized by security analysts to determine (a) the capability of a modeled system to deal with attacks and (b) the result of alterations of the system on its overall security.

A new game-theoretic model is introduced by [30] for the interaction between a network administrator and an attacker. The possible strategies of the attacker are illustrated using attack graphs, while the defender adds honeypots to the network to fool the attacker. By translating the attack graph into a Markov decision process (MDP) and employing a number of pruning techniques, the problems of realistic size are solved.

In a recent survey conducted by [31], various aspects of security models are compared and analyzed in terms of graphical security models (GrSM) phases, security metrics, and available tools. As a result, this survey can provide insight for users to decide the most appropriate GrSM to deal with their security concerns. Table 1 highlights the main characteristics of the existing schemes in attack graph modeling for CPSs as compared to our scheme.

The expressiveness of Halpern and Shoham's interval temporal logic (HS) is studied by [32] in the context of model checking (MC) in comparison with those of the standard point-based temporal logics (PTLs), linear temporal logic (LTL), computation tree logic (CTL), and CTL* (a superset of CTL). The results show that HS with trace-based semantics is equivalent to LTL, HS with computation-tree-based semantics is equivalent to finitary CTL*, and HS with state-based semantics is incomparable with LTL, CTL, and CTL*.

Several fixed parameter (FP) tractable cases are identified by [33] of the first order (FO) model checking problem of geometric graphs, and these are complemented by hardness results showing quite strict limits of FP tractability on the studied classes.

A new test statistic is established by [34] under the null hypothesis. Global and various local alternatives are presented to check the adequacy of the varying coefficient models when some covariate is measured with error. A behavior version is proposed by [35] of the annual data mining and knowledge discovery competition organized by association for computing machinery on Knowledge Discovery and Data Mining (KDD) CUP. Simulation experiments are carried out to evaluate the performance of the three model checking based algorithms, including LTL, interval temporal logic (ITL), and real time attack signature logic (RASL).

A method is proposed by [36] for model transformation from system modeling language SysML to new symbolic model checker/verifier (NuSMV) input language. The formal analysis method is adopted to verify and find defects from different aspects based on the NuSMV tool.

The recent literature study by [37] shows the state of the art methods for enhancing resilience of cyber-physical systems. Another classification of resilience enhancement depends on the resilience property (adaptation and recovery). The study also reviews the threats and the vulnerabilities that can affect the system's functionality.

A framework of labeled partial assignment interpolation systems (LPAIS) is presented by [38], which computes partial variable assignment interpolants (PVAIs) for propositional logic. The notion of logical strength is defined for LPAISs. The work shows how introducing a partial order over LPAISs allows for systematic comparison between the strength of the computed interpolants.

A multi-weighted extension is introduced by [39] to Kripke structures and CTL. The MC problem for the full logic is shown to be undecidable with three weights. However, by imposing upper-bounds on the temporal operators and assuming the cost converges over infinite runs, the synthesis problem is also decidable.

Table 1. Main characteristics of the existed schemes in attack graph modeling for cyber physical systems (CPSs).

Method	Aim	Proposed Solution	Merits	Demerits
Distributed attack graph generation [12]	Building vulnerability-based attack graphs on a distributed multi-agent platform.	Introduces a parallel and distributed memory-based algorithm for computation of attack graphs.	• Overcomes the state space explosion. • It can be utilized in real-time attack scenario detection and prediction.	Needs assessment of the advantages gained by allowing duplicate privilege expansion.
pwnPr3d [13]	Probabilistic threat modeling for automatic attack graph generation.	Generates probability distributions over the time to compromise assets.	The threat analysis is built-in and no security expertise is required.	A thorough experimentation on real-life systems is needed to validate the approach.
Attack tree [17]	Constructs attack trees to estimate the overall risk of a connected vehicle.	Uses graph transformation to model the car architecture and its state evolution under attacks.	Designed to support the conceptual phase of the vehicle's cyber-physical system.	Requires input data about structural and behavioral models of the service nodes, components, and the attacker model.
Attack graph modeling [19]	Attack graph for ambulatory medical devices.	Identifying vulnerabilities, assessing risk, and forming mitigation strategies when designing ambulatory devices.	The steps required to achieve an attack are easily identifiable.	More work is needed to consider the architecture and style of the attack graph.
Attack graph modeling [20]	Extends the Multi-host, Multi-stage Vulnerability Analysis (MulVAL) framework with a comprehensive network modeling.	Considers the network topology, short-range communication protocols, and their vulnerabilities.	• Supports short-range communication protocols. • Models specific industrial communication architectures. • Models various network attacks.	Further work is needed to support the automatic extraction of additional facts about wireless devices.
Safelite [22]	Constructing security modeling and analysis framework for networks.	Automatically converts an attack graph into a visualized hierarchical attack representation model (HARM).	Avoids state-space explosion problem.	• More security metrics are needed. • Further work is needed on the computation of the probability of attack success.
Bayesian attack graph (BAG) [23]	Models potential attack paths with the Bayesian network for power system.	Two BAG models are built to illustrate the attack procedures and to evaluate the probabilities of successful cyber-attacks.	Security countermeasures are implemented in the model to mitigate the damaging impacts of cyber-attacks.	A more comprehensive and realistic probabilistic model is needed.
Attack graph modeling [26]	Presents a framework of modeling and assessing security for the IoT.	Developed an Internot of Things (IoT) generator, a security model generator, and a security evaluator.	The framework is capable of mitigating potential attacks and addressing the scalability problem.	Security analysis is needed for introducing multiple targets, defense strategies, heterogeneity, and mobility.
Proposed scheme	Attack graph implementation and visualization for CPSs.	System model is checked using JKind and the generated attack scenarios are presented graphically.	The graphical user interface (GUI) visualizes the attack graph instead of long spread-sheets.	Requires overall specifications of the system model and the security property.

2. Nuclear Power Plant (NPP)

2.1. NPP System Architecture

Figure 1 illustrates a pressurized water NPP control system fitted from [40]. The following hierarchy can illustrate the system:

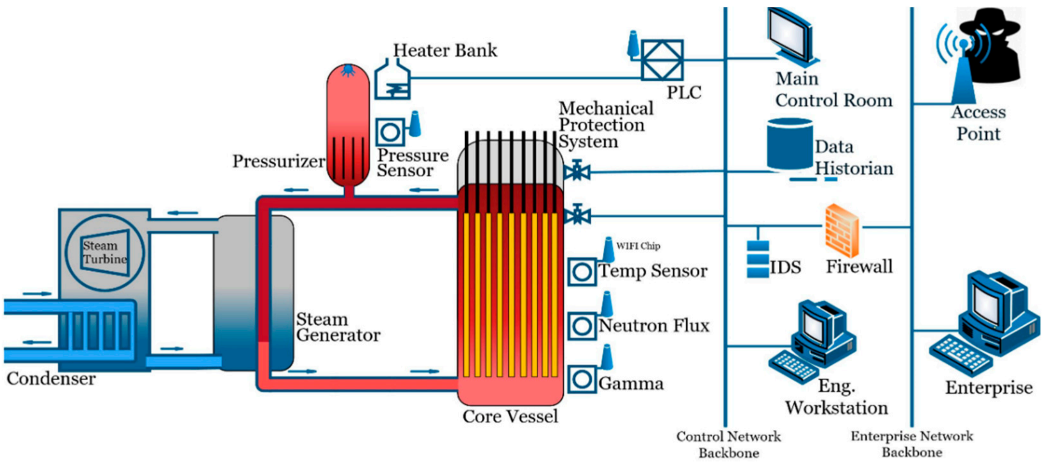


Figure 1. Pressurized water reactor.

Field Level, F: This level has three water loops—primary, secondary, and cooling. The primary loop is the main closed loop where the water is being heated and pressurized as a result of uranium-235 nuclear fission in the core vessel. Next, the heated water passes through a steam generator, thus transferring heat to the water in the secondary loop. Consecutively, the state of water is changed to steam. The steam loops until it reaches the steam turbine. Then, the steam's state is converted back to liquid using a condenser. This liquid loops back to the heat exchanger, thus closing the secondary loop. The cooling loop is either an open or a closed loop and is placed outside the plant.

The field level also includes safety and protection system, smart sensors (Sens) (e.g., temperature sensor, gamma sensor, neutron flux sensor, and pressure sensor), actuators (Acts), and a heater bank, which releases the heat during the day as required.

Control Level, C: This level consists of programmable logic controller (PLC) and remote terminal unit (RTU), which connect directly with devices from the field.

Supervisory Level, S: It is the main system, which receives and processes the digital data from C. The units of this level are: main control room (CR), data historian (DH*)*, and engineering workstation (EW).

Enterprise Level, E: It contains the enterprise site management computer (SM), which gathers data, sends results, and reports to decision makers, while a wireless access point (WA) [40] is placed for outside internet connection.

Network Backbones: Control network (CB) that links C with S and enterprise network (EB) that links S with E.

In addition, there is a firewall isolating the victims from the remaining enterprise network. An intrusion detection system (IDS) observes the network data stream. The firewall does not hold any admittance control restrictions on the flow of network data; rather, it lets the IDS observe data flow between (E;S), whereas the flow between (S;C) is not observed. For an attack instance/action that is detectable, the IDS activates an alarm upon its detection, while a stealthy attack stays undetected. SM, CR, DH, and EW have commercial off-the-shelf (COTS) OS vulnerability [41], while PLC, Sens, and Acts have firmware vulnerability [42].

The given vulnerabilities can be exploited, causing the successive possible attacks:

- Intelligent gathering (IG): This occurs when the attacker wants to gain more information (e.g., IP addresses, hardware model, OS COTS and firmware versions) and steals private data. This attack can be either stealthy or detectable.
- Malware injection (MI): This attack uses the operating software. When the attacker uses it, he/she can edit, copy, or install the code at the host. It can provide the attacker root access. This is a stealthy attack.
- Bypass security mechanism (BSM): This attack is utilized within a firmware update to attack a certain hardware (PLC, Sen) to obtain root access and gain control over it. This attack is detectable by IDS.
- Denial-of-Service (DoS): The attacker floods the server with requests for a certain service. This attack prevents service given to other users and creates latency. This attack is detectable by IDS [43].
- Man-in-the-Middle (MiM): It occurs when the attacker accesses a link (e.g., WIFI access point, sensor's WIFI chip) between two parties, giving the attacker root access on the communication transmitter (e.g., stealing and/or generating illegitimate traffic). This attack is detectable by IDS.
- Alteration-of-Data (AoD): It takes place if the attacker has access to a software. It targets the device's memory to generate data processing latency and data alteration (e.g., changing the pressure set point).

2.2. Formal NPP Depiction

The system can be formally characterized as follows:

1. The attacker is assumed to be placed at *WA* and has *root* privilege (static).
2. Set of field level elements F; variable $f \in \{Sen, Act\}$ (static).
3. Set of control level elements C; variable $c \in \{PLC, RTU\}$ (static).
4. Set of supervisory level elements S; variable $s \in \{CR, DH, CB, EW\}$ (static).
5. Set of enterprise level elements E; variable $e \in \{WA, SM\}$ (static).
6. System connectivity, $Z \subseteq E \times E, E \times S, S \times S, S \times C, C \times F$; $z_{ij} = 1$ iff element i is connected to element j (static).
7. System vulnerabilities V; Boolean $v_i = 1$ iff vulnerability $v \in \{COTS, firmware\}$ is placed on host i (static).
8. Set of possible attacks A; variable $a \in \{IG, MI, BSM, DoS, MiM, AoD\}$ (static).
9. Attack instances, $AI \subseteq A \times (E \times E, E \times S, S \times S, S \times C, C \times F)$; labeled $a_{ij} \equiv$ attack a from source i to target j, $a \in A$ (static).
10. Attacker level of privilege P on machine/host $i \in \{S, E\}$; variable $p_i \in \{\text{none, user, root}\}$ (dynamic).
11. Hardware control H on device $i \in \{F, C\}$; Boolean $h_i = 1$ iff attacker gains control over the firmware of device i (dynamic).
12. Latency L from element i; Boolean $l_i = 1$ iff communication from i is delayed (dynamic).
13. Input data N into host i; Boolean $n_i = 1$ iff input of element i is corrupted (dynamic).
14. Output data O from host i; Boolean $o_i = 1$ iff output of elemet i is corrupted (dynamic).
15. Data knowledge K; Boolean $k_j = 1$ iff attacker gets knowledge from j (dynamic).
16. Intrusion detection system *IDS*: $A \times E \times S \rightarrow \{0,1\}$; Boolean $\text{ids}(a_{ij}) = 1$ iff attack a from source i to target j is detectable (static).
17. A global Boolean dg tracks whether an IDS alarm has been triggered for any earlier executed atomic attack (dynamic).
18. Attack instances/actions pre-conditions:

    - $Pre(IG_{ij}) \equiv (z_{ij} = 1) \wedge (p_i \geq user)$.
    - $Pre(MI_{ij}) \equiv (z_{ij} = 1) \wedge (p_i \geq user) \wedge (COTS_j = 1) \wedge (\exists y \in \{S, E\}: k_y = 1)$.
    - $Pre(BSM_{ij}) \equiv (z_{ij} = 1) \wedge (p_i \geq \text{user} \vee (h_i = 1)) \wedge (firmware_j = 1) \wedge (\exists y \in \{F, C\}: k_y = 1)$.

- $Pre(DoS_{ij}) \equiv (z_{ij} = 1) \wedge (p_i = root \vee (h_i = 1)) \wedge (COTS_j = 1 \vee (firmware_j = 1))$.
- $Pre(MiM_{ij}) \equiv (z_{ij} = 1) \wedge (p_i = root \vee (h_i = 1))$.
- $Pre(AoD_{ij}) \equiv (z_{ij} = 1) \wedge (p_i = root \vee (h_i = 1))$.

19. Attack instances/actions post-conditions:

- $Post(IG_{ij}) \equiv k_j = 1 \wedge ((i = SM \vee WA) \wedge (dg = 0) \Rightarrow (dg = 0) \vee (dg = 1))$.
- $Post(MI_{ij}) \equiv p_j = root$.
- $Post(BSM_{ij}) \equiv h_j = 1 \wedge ((i = SM \vee WA) \wedge (dg = 0) \Rightarrow dg = 1)$.
- $Post(DoS_{ij}) \equiv l_j = 1 \wedge k_j = 1 \wedge ((i = SM \vee WA) \wedge (dg = 0) \Rightarrow dg = 1)$.
- $Post(MiM_{ij}) \equiv o_j = 1 \wedge k_j = 1 \wedge ((i = SM \vee WA) \wedge (dg = 0) \Rightarrow dg = 1)$.
- $Post(AoD_{ij}) \equiv l_j = 1 \wedge n_j = 1$.

20. Initial state: $pWA = root \wedge (\forall j \in \{F, C, S, E\}$: $Pj = \text{none} \wedge (lj = hj = kj = nj = oj = 0)) \wedge dg = 0$.
21. *Security property* φ is that attacker cannot disrupt the *NPP*. This property can then be expressed by CTL as:

$$\varphi \equiv \mathrm{AG}((\mathrm{lPLC} = 0) \wedge (\mathrm{hPLC} = 0) \vee (\mathrm{dg} = 1)) \equiv \mathrm{AG}(\neg((\mathrm{lPLC} = 1) \vee (\mathrm{hPLC} = 1) \wedge (\mathrm{dg} = 0))).$$

3. Implementation of Cyber-Attack Scenarios

The model-based technique for attack graph implementation demands a formal model of the system and security property of concern. The following definitions are adapted from [44] to formally define an attack graph.

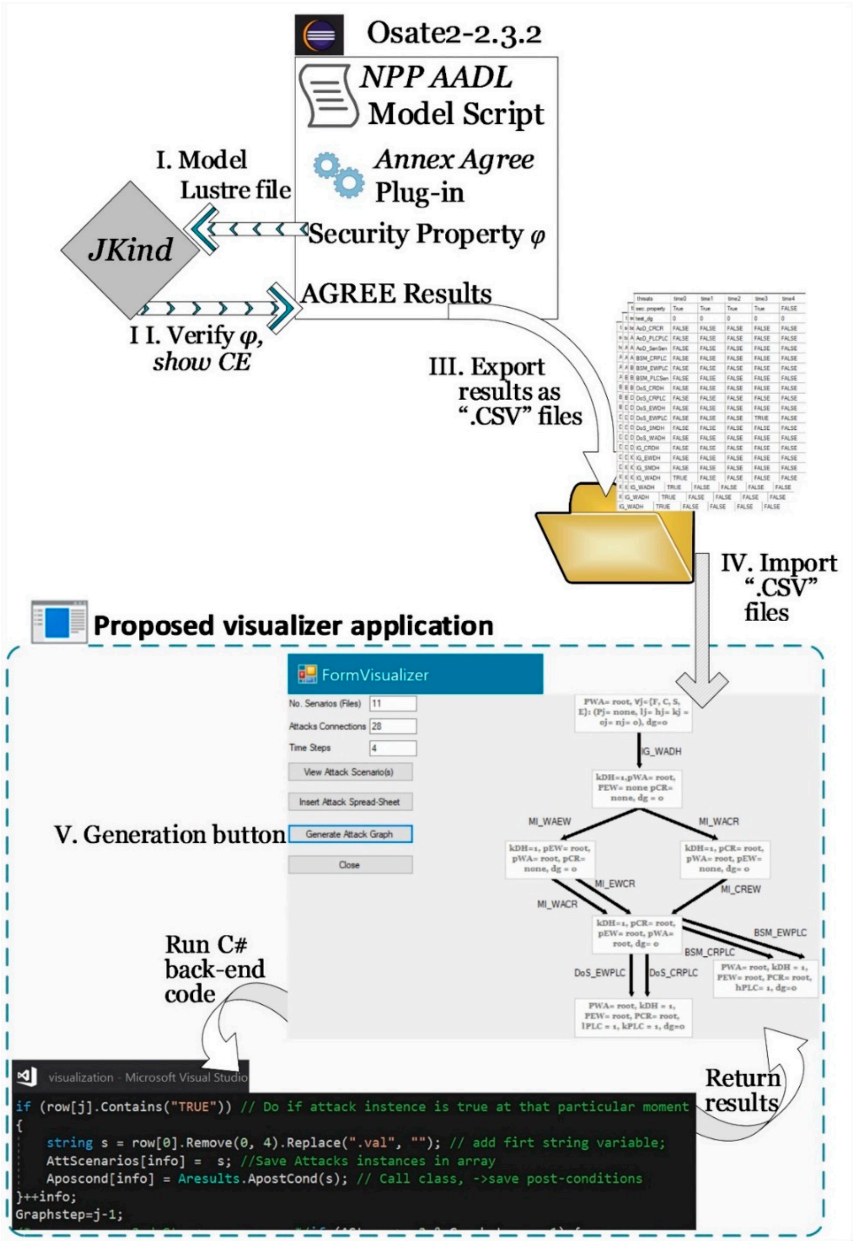
Definition 1. *A system security model ($M = (S, E, s_0)$) is a state-transition diagram whose locations S, with $s_0 \in S$ defining an initial location, identify the security status of the system, and whose transitions E illustrate how the attack instances cause a change in the system security status. Overall, the transitions are determined by pre-conditions on state-variables, and their execution apply certain post-conditions on the same state-variables.*

Definition 2. *Given a security model M and a security property φ, an attack path (AP) is a finite acyclic path of a sequence of states in M, $AP = (s_0, s_1, \ldots, s_f)$, where s_0 is an initial state in M, while any two adjacent states in the path belong to the transition set E, such that the execution of AP leads to the violation of φ.*

Definition 3. *An attack graph (AG) is a data structure illustrating a union of all attack paths.*

$$\mathrm{AG} = \{\mathrm{AP_i} | \mathrm{AP_i} \text{ is an attack path in M}\} \quad (1)$$

Two software tools are used to initiate the cyber-attack scenarios generation and visualization, as shown in Figure 2. These tools are JKind model checker [45] and Microsoft Visual Studio [46]. JKind is an infinite state model checker for verifying safety properties of synchronous systems [47], which are written in the Lustre, a formally determined, declarative, and synchronous dataflow programming language for programming reactive systems [48]. The checking is based on k-induction and property directed reachability using a back-end satisfiability modulo theories (SMT) solver. A checked property is true for all executions of the system. A wrong property is given with a definite counter-example (CE) illustrating the property violation, which is presented here as an attack scenario given as a sequence of attack instances causing system compromise.


Osate2-2.3.2
NPP AADL Model Script
Annex Agree Plug-in
Security Property φ
AGREE Results
I. Model Lustre file
JKind
I I. Verify φ, show CE
III. Export results as ".CSV" files
IV. Import ".CSV" files
Proposed visualizer application
FormVisualizer
View Attack Scenario(s)
Insert Attack Spread-Sheet
Generate Attack Graph
Close
V. Generation button
PWA= root, ∀j∈{F, C, S, E}: (Pj= none, lj= hj= kj = oj= nj= o), dg=o
IG_WADH
kDH=1,pWA= root, PEW= none pCR= none, dg = o
MI_WAEW
MI_WACR
kDH=1, pEW= root, pWA= root, pCR= none, dg = o
kDH=1, pCR= root, pWA= root, pEW= none, dg= o
MI_EWCR
MI_WACR
MI_CREW
kDH=1, pCR= root, pEW= root, pWA= root, dg= o
BSM_EWPLC
BSM_CRPLC
DoS_EWPLC
DoS_CRPLC
PWA= root, kDH = 1, PEW= root, PCR= root, hPLC= 1, dg=o
PWA= root, kDH = 1, PEW= root, PCR= root, lPLC = 1, kPLC = 1, dg=o
Run C# back-end code
Return results
visualization - Microsoft Visual Studio

threats	time0	time1	time2	time3	time4
sec. property	True	True	True	True	FALSE
test_dg	0	0	0	0	0
AoD_CRCR	FALSE	FALSE	FALSE	FALSE	FALSE
AoD_PLCPLC	FALSE	FALSE	FALSE	FALSE	FALSE
AoD_SenSen	FALSE	FALSE	FALSE	FALSE	FALSE
BSM_CRPLC	FALSE	FALSE	FALSE	FALSE	FALSE
BSM_EWPLC	FALSE	FALSE	FALSE	FALSE	FALSE
BSM_PLCSen	FALSE	FALSE	FALSE	FALSE	FALSE
DoS_CRDH	FALSE	FALSE	FALSE	FALSE	FALSE
DoS_CRPLC	FALSE	FALSE	FALSE	FALSE	FALSE
DoS_EWDH	FALSE	FALSE	FALSE	FALSE	FALSE
DoS_EWPLC	FALSE	FALSE	FALSE	TRUE	FALSE
DoS_SMDH	FALSE	FALSE	FALSE	FALSE	FALSE
DoS_WADH	FALSE	FALSE	FALSE	FALSE	FALSE
IG_CRDH	FALSE	FALSE	FALSE	FALSE	FALSE
IG_EWDH	FALSE	FALSE	FALSE	FALSE	FALSE
IG_SMDH	FALSE	FALSE	FALSE	FALSE	FALSE
IG_WADH	TRUE	FALSE	FALSE	FALSE	FALSE
IG_WADH	TRUE	FALSE	FALSE	FALSE	FALSE
IG_WADH	TRUE	FALSE	FALSE	FALSE	FALSE
IG_WADH	TRUE	FALSE	FALSE	FALSE	FALSE

No. Senarios (Files)	11
Attacks Connections	28
Time Steps	4

```
if (row[j].Contains("TRUE")) // Do if attack instence is true at that particular moment
{
    string s = row[0].Remove(0, 4).Replace(".val", ""); // add firt string variable;
    AttScenarios[info] =  s; //Save Attacks instances in array
    Aposcond[info] = Aresults.ApostCond(s); // Call class, ->save post-conditions
}++info;
Graphstep=j-1;
```

Figure 2. Cyber-attack scenarios generation workflow.

In this work, system specification models of units and their interfaces and connections are encoded using AADL within the open-source integrated development environment (Osate2) [8]. The AADL architecture model is embedded by AGREE Annex plug-in [49] that is used to identify the units' structures and system-level security properties. AGREE also converts the AADL plus Annex models and properties to Lustre and connects with JKind, which checks the system with a security property of concern φ and submits the result as if a CE exists.

Once all CEs (attack scenarios) are generated, we can export them as comma-separated values (CSV) files into a *visualizer* Windows application that creates a GUI, which is encoded using C# within Microsoft Visual Studio. Visual Studio is an integrated development environment (IDE) that utilizes the Windows platform to implement programs, websites, as well as graphical visualization of data [46].

The primary encoded GUI interface has three major features: CEs' attributes, actions, and results. In CEs' attributes, the number of CSV files (CEs) to be read is listed in "*No. Scenarios (Files)*". The "*Attack Connections*" field defines all potential attack instances between the model's components.

"*Time Steps*" field defines the maximum number of time steps (transitions between attack instances in the attack scenarios) to be presented. Once these fields are defined, attack scenario(s) can be visualized by selecting "*View Attack Scenario(s)*", "*Insert Attack Spread-Sheet*", and "*Generate Attack Graph*" options. The results illustrates the CE spreadsheet viewer, attack scenarios, and final state post-conditions.

To execute this GUI for the NPP, we define the given security property φ, where the intention of the attacker is to disrupt the system by either creating a denial of service from PLC (shown by its latency) or obtaining control on PLC with no detection. This goal can be achieved by obtaining root privilege on CR and EW machines, respectively. Therefore, the property φ that must not be breached is that either the attacker never produces system disruption or the attacker is detected by the IDS. The JKind produced the counter-example/attack-scenario $CE1 := IG_{WADH} \rightarrow MI_{WAEW} \rightarrow MI_{EWCR} \rightarrow DoS_{EWPLC}$ as a spread sheet with 205 rows, which was then stored as a separate CSV file. Having produced all 16 CEs CSV files, the number of CEs to be visualized is listed in "*No. Scenarios (Files)*" field. The number of attack instances/actions is 28 (these instances are encoded in the AADL model). In addition to that, the number of time steps is four. Then, selecting "*View Attack Scenario(s)*", the generated CE1 can be presented in the GUI.

This attack scenario is explained as follows. First, the attacker has root privilege on WA, and an IG_{WADH} attack is conducted to gain knowledge on the system and its units (e.g., IP addresses and OS). Employing the discovered information, a MI_{WAEW} attack is conducted, exploiting a COTS vulnerability in EW. This attack aims to breach the computer OS by obtaining root access. At this stage, there are various attack instances to get on with (i.e., the pre-conditions of BSM_{EWPLC}, MI_{EWCR}, and DoS_{EWDH} are satisfied). However, in this scenario, a MI_{EWCR} is conducted to obtain root access on the control machine CR. Next, there is a DoS_{EWPLC} attack against PLC to cause latency and prevent other machines from demanding any information from PLC.

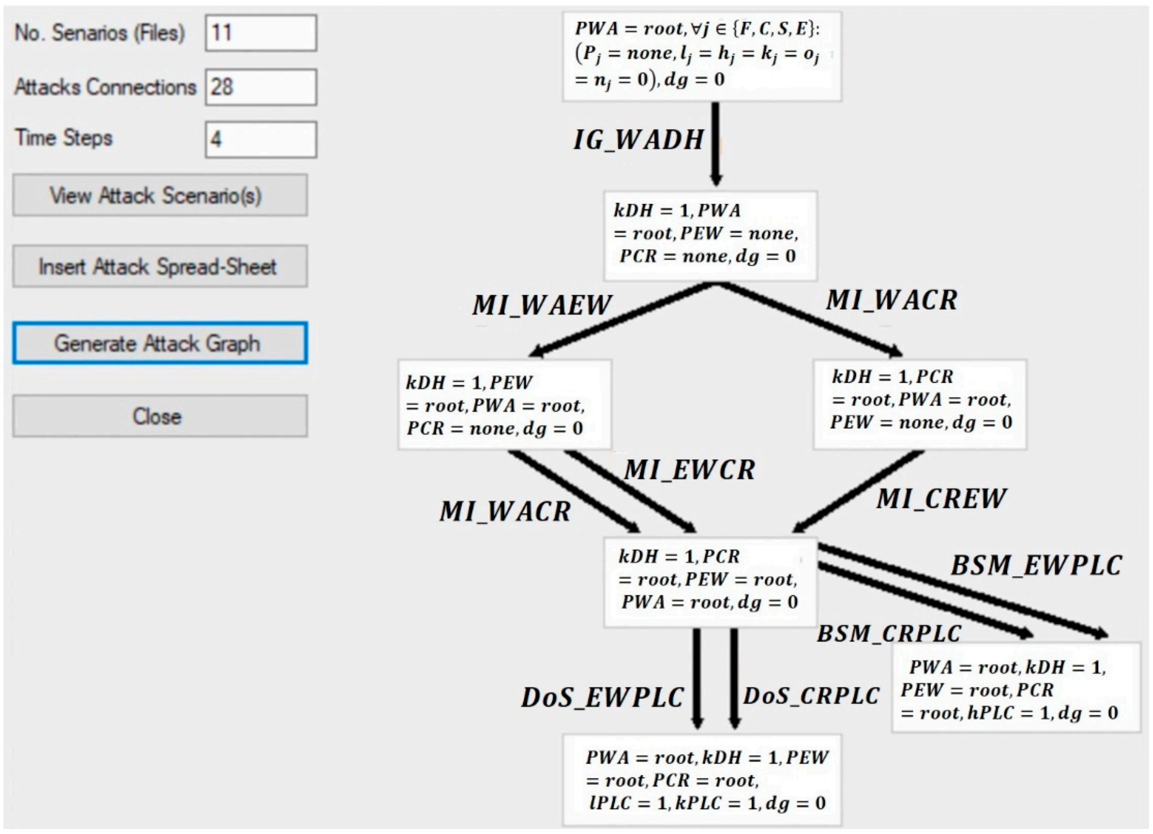
By encoding this counterexample CE1 in disjunct with the property φ being verified, i.e., $\varphi \vee CE1$, a new counterexample fulfills $\neg(\varphi \vee CE1) = \neg\varphi \wedge \neg CE1$, i.e., a counterexample of φ and not CE1. This generates a new counterexample $CE2: = IG_{WADH} \rightarrow MI_{WACR} \rightarrow MI_{WAEW} \rightarrow BSM_{CRPLC}$. In this scenario, after obtaining a root privilege on both EW and CR machines, BSM_{EWPLC} attack uses firmware vulnerability of PLC to obtain control over it. CE2 is given by JKind as a spreadsheet with 206 rows. By doing this procedure, many (but still finite in number) CEs can be produced, resulting in all attack scenarios, i.e., the "attack graph".

It can be noticed that the number of rows in the spreadsheets generated by JKind increase by one for every newly generated CE (e.g., CE16 spreadsheet has 220 rows). To manage the number of rows in the produced CEs and to represent them in an appealing graphical way, the visualizer Windows application extracts only the attack scenarios from the generated CSV files and presents them in the GUI.

To illustrate a specific CE as a spreadsheet, the "*Insert Attack Spread-Sheet*" action can be selected, and the sheet can be visualized in the spreadsheet viewer. The resulted spreadsheet demonstrates a possible attack sequence CE1. This sequence consists of four attack instances such that one attack instance can occur at each time step. This sheet has 28 rows, which reduces the total number of rows by 177 as compared to the sheet generated by JKind.

By selecting "*Generate Attack Graph*", the set of attack scenarios violating the property φ resulting in NPP system disruption is given by its attack graph, as illustrated in Figure 3. This attack graph has 16 attack scenarios that lead to two terminating states.

In this graph, it can be noted that the *DoS* and the *MI* attack instances show in every attack scenario as a result of COTS vulnerability. Hence, if the assets can be used to improve the machines' OS and alleviate this vulnerability, then the security can be enhanced. We can also observe that PLC can be abused by an attacker using the available firmware vulnerability to disrupt the system. Hence, in the design-phase, the system designers can propose suitable improvements as suggested by [50] to enhance the system-level security characteristics.



No. Senarios (Files) 11

Attacks Connections 28

Time Steps 4

View Attack Scenario(s)
Insert Attack Spread-Sheet
Generate Attack Graph
Close
$PWA = root, \forall j \in \{F, C, S, E\}$: $(P_j = none, l_j = h_j = k_j = o_j = n_j = 0), dg = 0$
IG_WADH
$kDH = 1, PWA = root, PEW = none, PCR = none, dg = 0$
MI_WAEW
MI_WACR
$kDH = 1, PEW = root, PWA = root, PCR = none, dg = 0$
$kDH = 1, PCR = root, PWA = root, PEW = none, dg = 0$
MI_EWCR
MI_WACR
MI_CREW
$kDH = 1, PCR = root, PEW = root, PWA = root, dg = 0$
BSM_EWPLC
BSM_CRPLC
$PWA = root, kDH = 1, PEW = root, PCR = root, hPLC = 1, dg = 0$
DoS_EWPLC
DoS_CRPLC
$PWA = root, kDH = 1, PEW = root, PCR = root, lPLC = 1, kPLC = 1, dg = 0$

Figure 3. Nuclear power plant (NPP) generated attack graph.

4. Industrial Control System (ICS)

4.1. *ICS Topology*

Figure 4 shows an ICS with the following components.

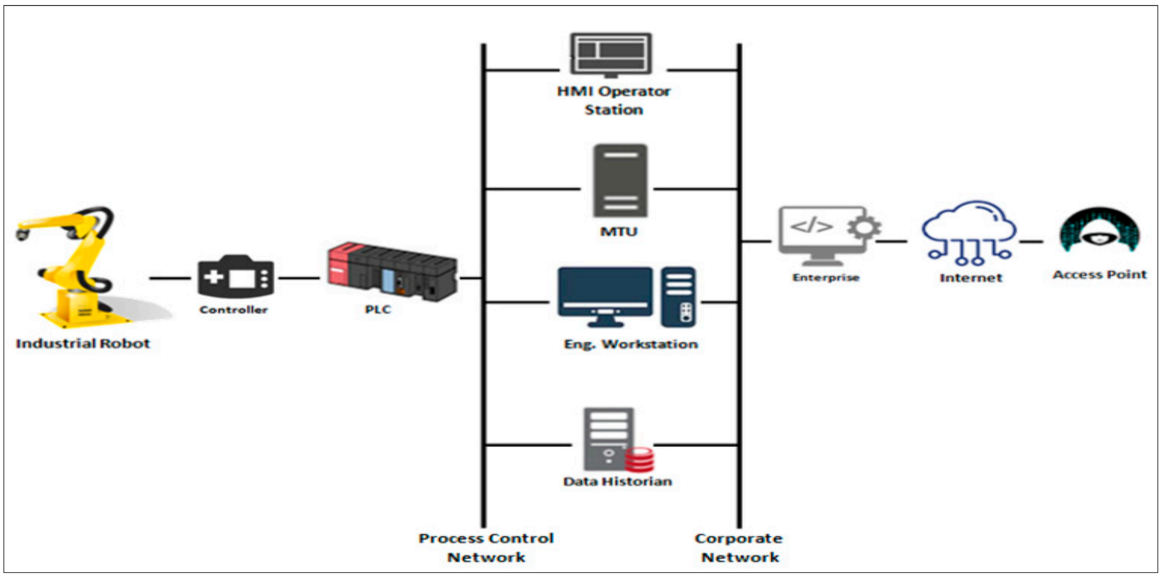


Figure 4. Industrial control system (ICS).

Physical Level (P): This level includes a PLC and a micro-controller (MC) that communicate directly with the industrial robot. These elements also conduct logical processes and data transfer to the supervisory control and data acquisition (SCADA) level. The industrial robot (IR) includes multiple hardware and software components such as mechanical actuators, control logic, and operating systems.

SCADA Level (S): This is the main component that receives and processes the digital data from the physical level. It includes human machine interface (HMI), DH, EW, and master terminal unit (MTU).

Corporate Level (C): This level includes enterprise site management computer (E), which gathers data, sends results, and reports to decision makers, while wireless access point (AP) exists for external internet communication.

Network Backbones: Process control network that links S with P and corporate network that links C with S.

For the ICS system, two vulnerabilities are considered: COTS and firmware vulnerability on PLC.

4.2. Possible Attacks Against ICS

The illustrated vulnerabilities can be exploited resulting in the following attacks:

- Social Engineering (SE): This attack generally targets enterprises and organizations; an attacker uses a human to get a critical information by psychologically manipulating the user or deceiving him [51].
- Malware Injection (MI).
- Pivoting (P): This attack generally consists of two steps, phishing and malware injection. This attack is usually used on the enterprise network as it has the most visibility to the Internet. Compromising this network gives the attacker more access into the system and down to the process control domain [52].
- Sniffing (S): In network security, it corresponds to stealing or breaking off data by capturing the network traffic by a sniffer [53].
- Buffer Overflow (BO): This attack is used to fuzz the SCADA HMI and break it down due to a heap buffer overflow. In addition, the attacker can perform preferred payloads on the HMI server such as running a remote shell [53].
- Privilege Escalation (PE): This attack relays on programming errors or design flaws to give the attacker more access to the network and its associated data and applications [54].
- Authentication Bypass (B): This attack allows the attacker to gain system or network access, bypass a mechanism, a flaw in the design, or an alternative access path placed by developers.
- Man-in-the-Middle (MITM).
- Alteration-of-Data (AoD).
- Denial-of-Service (DoS).

4.3. ICS Formal Depiction

The system can be formally described as follows:

1. The attacker is assumed to be placed at AP and has root privilege.
2. Set of corporate level elements C; variable $c \in \{E, AP\}$ (static).
3. Set of SCADA level elements S; variable $s \in \{HMI, MTU, DH, EW\}$ (static).
4. Set of physical level elements P; variable $p \in \{PLC, C, IR\}$ (static).
5. System connectivity, $N \subseteq P \times P, P \times S, S \times S, S \times C, C \times C$; $n_{ij} = 1$ if element i is connected to element j (static).
6. System vulnerabilities V; Boolean $v_i = 1$ if vulnerability $v \in \{firmware, COTS\}$ is placed on host i (static).
7. Set of possible attacks A, variable $a \in \{MI, SE, P, S, BO, PE, B, DoS, AoD, MITM\}$.
8. Attack instances, $AI \subseteq \mathrm{A} \times (P \times P, P \times S, S \times S, S \times C, C \times C)$; labeled $a_{ij} \equiv$ attack a from source i to target j.
9. Attacker level of authority Y on machine/host $I \in \{P, S, C\}$; variable $y_i \in \{none, user, root\}$ (dynamic).

10. Data knowledge K; Boolean $k_j = 1$ if attacker gets knowledge about j (dynamic).
11. Delay D from element i; Boolean $d_i = 1$ if communication from i is delayed (dynamic).
12. Corruption of data O, the data coming from source i to target j is corrupted; Boolean $o_j = 1$ (dynamic).
13. Attack pre-conditions:
    - Pre $(SE_{ij}) \equiv (n_{ij} = 1) \wedge (y_i \geq root)$
    - Pre $(P_{ij}) \equiv (n_{ij} = 1) \wedge (y_i \geq root) \wedge (COTS_j = 1)$
    - Pre $(PE_{ij}) \equiv (n_{ij} = 1) \wedge (y_i \geq user)$
    - Pre $(DoS_{ij}) \equiv (n_{ij} = 1) \wedge (y_i = root) \wedge (COTS_j = 1 \vee firmware_j = 1) \wedge (\exists\, e \in \{P, S\}{:}k_e = 1)$
    - Pre $(S_{ij}) \equiv (n_{ij} = 1) \wedge (y_i = root)$
    - Pre $(BO_{ij}) \equiv (n_{ij} = 1) \wedge (y_i = root)$
    - Pre $(MI_{ij}) \equiv (n_{ij} = 1) \wedge (y_i \geq \text{user}) \wedge (COTS_j = 1) \wedge (\exists\, e \in \{S\}{:}k_e = 1)$
    - Pre $(B_{ij}) \equiv (n_{ij} = 1) \wedge (y_i = user) \wedge (COTS_j = 1 \vee firmware_j = 1) \wedge (\exists\, e \in \{P, S\}{:}k_e = 1)$
    - Pre $(AoD_{ij}) \equiv (n_{ij} = 1) \wedge (y_i \geq user)$
    - Pre $(MITM_{ij}) \equiv (n_{ij} = 1) \wedge (y_i \geq user)$
14. Attack post-conditions:
    - Post $(SE_{ij}) \equiv (k_j = 1) \wedge (y_j = user)$
    - Post $(P_{ij}) \equiv (y_j = root)$
    - Post $(PE_{ij}) \equiv (y_j = root)$
    - Post $(DoS_{ij}) \equiv (y_j = user) \wedge (k_j = 1) \wedge (d_j = 1)$
    - Post $(S_{ij}) \equiv (y_j = root) \wedge (k_j = 1)$
    - Post $(BO_{ij}) \equiv (y_j = root) \wedge (d_j = 1)$
    - Post $(MI_{ij}) \equiv (y_j = root)$
    - Post $(B_{ij}) \equiv (y_j = root)$
    - Post $(AoD_{ij}) \equiv (d_j = 1) \wedge (o_j = 1) \wedge (y_j = root)$
    - Post $(MITM_{ij}) \equiv (d_j = 1) \wedge (o_j = 1) \wedge (y_j = root)$
15. Initial state:

    $(y_{AP} = root) \wedge (\forall\, j \in \{P, S, T\}{:}\ (y_j = none) \wedge (k_j = o_i = d_j = 0))$
16. Security property (φ): is the attacker cannot disrupt the system (i.e., no delay from IR, no change in data entering IR). This property can be then described CTL as:

    $\varphi \equiv \text{AG}\ (d_{IR} = 0) \wedge (o_{IR} = 0) \equiv \text{AG}\ (\neg\ ((d_{IR} = 1) \vee (o_{IR} = 1)))$

4.4. ICS Attack Scenarios Generation

Considering the given security property φ, in which the attacker goal is to cause system disruption by either inducing a delay from IR or changing the data entering IR, the JKind model checker generates the following counter-example (*CE1:SE_APE → BO_EHMI → B_HMIPLC → AoD_PLCMC → MITM_MCIR*) as a spreadsheet. This sheet has five attack instances such that one attack instance can occur at every time step. This attack sequence is illustrated as follows. First, the attacker has root access on AP; an *SE_APE* attack is conducted to collect information about the system and its elements and to gain access to the enterprise. Using the disclosed information and the access to the enterprise, a *BO_EHMI* attack is launched against the HMI to gain root access on it. Next, a *B_HMIPLC* attack is conducted, exploiting COTS or firmware vulnerability in PLC. Then, an *AoD_PLCMC* is conducted to gain root access on the MC. Next, an *MITM_MCIR* attack against IR occurs to cause data corruption entering the industrial robot (e.g., changing IR set point). In addition, it causes delay on the IR performance.

By encoding this generated counter example CE1 in disjunct with the property φ being checked, which is ($\varphi \vee$ *CE1*), a new counter example of φ that is different from CE1 is generated. By continuing this process, six CEs are discovered, yielding the complete attack graph as shown in Figure 5. From the resulting graph, it can be noted that the PLC is the most compromised component of the system, as it mainly controls the industrial robot. By placing an intrusion detection system (IDS) [55] between the corporate and the SCADA levels of the ICS model to monitor the network traffic flow, it is seen that no property violation occurs, thus reducing the attack graph.

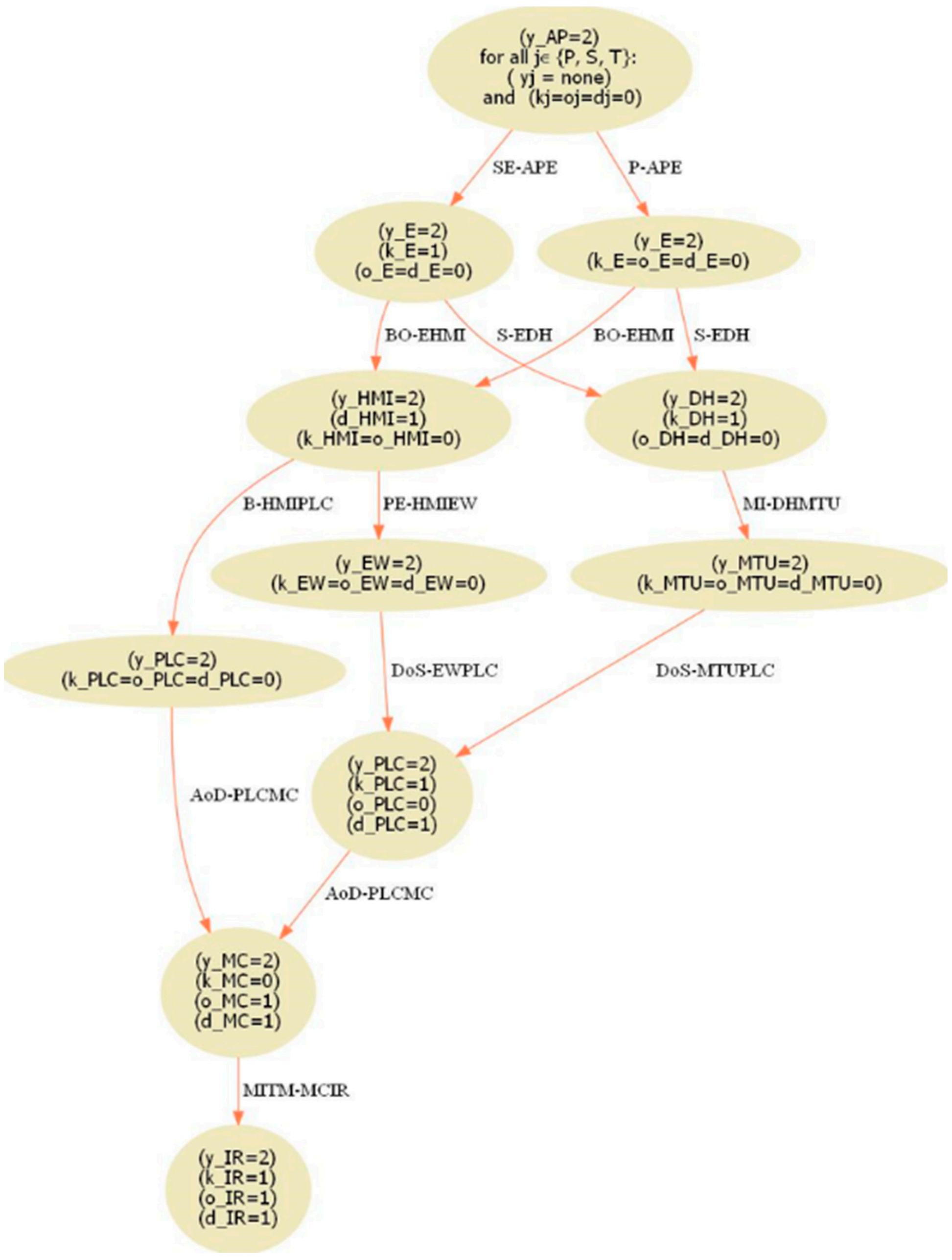

(y_AP=2) for all j∈ {P, S, T}: (yj = none) and (kj=oj=dj=0)
SE-APE
P-APE
(y_E=2) (k_E=1) (o_E=d_E=0)
(y_E=2) (k_E=o_E=d_E=0)
BO-EHMI
S-EDH
BO-EHMI
S-EDH
(y_HMI=2) (d_HMI=1) (k_HMI=o_HMI=0)
(y_DH=2) (k_DH=1) (o_DH=d_DH=0)
B-HMIPLC
PE-HMIEW
MI-DHMTU
(y_EW=2) (k_EW=o_EW=d_EW=0)
(y_MTU=2) (k_MTU=o_MTU=d_MTU=0)
(y_PLC=2) (k_PLC=o_PLC=d_PLC=0)
DoS-EWPLC
DoS-MTUPLC
AoD-PLCMC
(y_PLC=2) (k_PLC=1) (o_PLC=0) (d_PLC=1)
AoD-PLCMC
(y_MC=2) (k_MC=0) (o_MC=1) (d_MC=1)
MITM-MCIR
(y_IR=2) (k_IR=1) (o_IR=1) (d_IR=1)

Figure 5. ICS attack graph.

5. Vehicular Network System (VNS)

5.1. VNS Topology

Figure 6 shows the vehicular network. This network can be illustrated as follows:

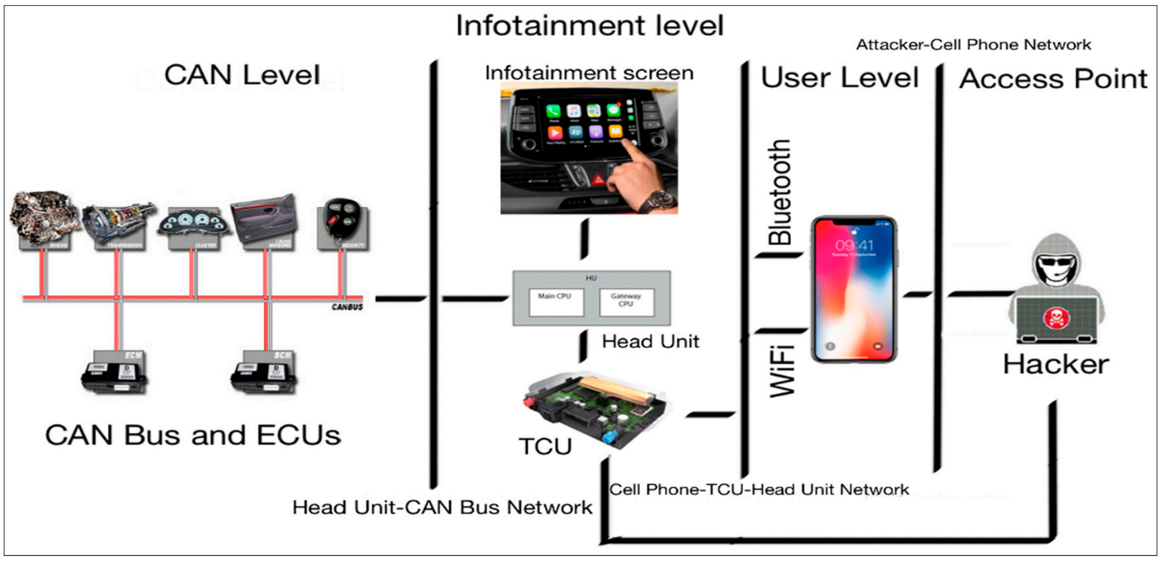


Figure 6. Vehicular network system (VNS) architecture.

Access Point (AP): It exists for outside internet communication. We assume the attacker is located at this point.

User Level (UL): It is the point where the user of the vehicle uses his/her personal mobile phone and connects it to the car's Bluetooth or the WiFi hotspot given off by the vehicle.

Infotainment Level (IL): It is made up of the infotainment screen, the head unit that contains the gateway CPU and the main CPU, and the telematics unit.

Controller Area Network (CAN) Level: It contains the CAN bus and the ECUs connected to the bus.

For the CAN bus, three vulnerabilities are identified:

- Resource Exhaustion (RE): It occurs when the resources required to perform an action are entirely expended.
- Lack of Authentication (LoA): This allows the attacker to access the CAN bus pretending to be someone he/she is not [56].
- Multicast Messaging (MM): When a message is sent to the CAN bus, it has no certain destination. Each access point or controller on the bus has access to all the messages [56].

5.2. Possible Attacks Against VNS

The presented vulnerabilities can be exploited resulting in the following attacks:

- Trojan-Horse (TH): The attacker develops a malicious app that allows the attacker to obtain access to the user's phone. This allows the attacker to exploit the Bluetooth connection of the driver's phone to the infotainment gateway [57].
- Packet Sniffing (PS).
- Client Side Attack (CSA): Using this attack, the attacker can take over the user's web browser of the vehicle by injecting a scripted malware [58].
- Buffer Overflow (BO).
- Privilege Escalation (PE).
- Row Hammer (RH): An attacker can execute a malicious application that continuously alters the same row of transistors in a fraction of a second, creating a way for the attacker to gain control over the victim's device [59].
- Tampering with Data (TwD).
- Denial-of-Service (DoS).

- Dictionary Attack (DA): This attack uses a large set of words to generate potential passwords.
- Brute Force Attack (BF): This attack uses automated software to conduct a large number of suggested values for the desired data, such as passwords.
- Hybrid Attack (HA): Dictionary attack and Brute Force attack are used together.
- Cryptographic Attack (CA): This attack consists of listening in on information being traded between two parties and modifying the information without their knowledge.
- CAN Dump (CD): It can be used to dump all the traffic on the CAN bus on to the attacker's terminal [60].
- CAN Sniffer (CS): The attacker looks at the traffic for specific message IDs and begins to filter repeated messages that do not change [61].
- CDS: CAN dump and CAN sniffer are used together.
- Session Hijacking (SH): An attacker uses a sniffer to look through authentic requests and finds the information needed to impersonate the authentic user [62].
- Eavesdropping (MIM).

5.3. VNS Formal Depiction

The VN can be formally described as follows:

1. An attacker is considered to be placed at the AP and has root access over it.
2. Set of ECUs Z; variable $z \in \{BCM, ECM, ACC, EPB, SCCM, PSCM\}$; Boolean $k_j = 1$ if attacker gets knowledge about j (static).
3. Response R from element i; Boolean $r_i = 0$ if element i does not respond (dynamic).
4. CAN bus level C (static).
5. Set of possible attacks A, variable $a \in \{RH, CA, HA, PE, CDS, TH, CSA, DoS, MiM, DA, BF, BO, SH, TwD\}$.
6. Attack instances, $AI \subseteq A \times (U \times U, U \times F, F \times F, F \times C, C \times Z)$; labeled $a_{ij} \equiv$ attack a from source i to target j.
7. Set of infotainment level elements F; variable $f \in \{TCU, HU\}$ (static).
8. Set of user level elements U; variable $u \in \{AP, CP\}$, where CP is cell phone device (static).
9. System connectivity, $\mathrm{Q} \subseteq \mathrm{U} \times \mathrm{U}, \mathrm{U} \times \mathrm{F}, \mathrm{F} \times \mathrm{F}, \mathrm{F} \times \mathrm{C}, \mathrm{C} \times \mathrm{Z}$; $q_{ij} = 1$ if element i connected to element j (static).
10. System vulnerabilities V; Boolean $v_i = 1$ if vulnerability $\mathrm{v} \in \{\mathrm{RE, LoA, MM}\}$ is located on host i (static).
11. Attacker level of privilege P on machine/host $\mathrm{i} \in \{\mathrm{Z, C, F, U}\}$; variable pi $\in$ {none, user, root} (dynamic).
12. Level of danger D on vehicle; variable d $\in$ {none, low, high} (dynamic).
13. Attack pre-conditions:

    - Pre$(RH_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = none)$
    - Pre$(TH_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = root)$
    - Pre$(CA_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = root)$
    - Pre$(HAij) \equiv (q_{ij} = 1) \wedge (p_i = root)$
    - Pre$(MiM_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = user)$
    - Pre$(CSA_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = user)$
    - Pre $(BO_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = user)$
    - Pre $(CDS_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = \mathrm{user})$
    - Pre $(DoS_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = root) \wedge (v_j = 1) \wedge (r_j = 1) \wedge (k_j = 1)$
    - Pre $(SH_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = user)$
    - Pre $(PE_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = none)$
    - Pre $(TwD_{ij}) \equiv (q_{ij} = 1) \wedge (p_i = root)$

14. Attack post-conditions:

- Post $(RH_{ij}) \equiv (p_j = root) \wedge (d = none)$
- Post $(\mathrm{TH}_{ij}) \equiv (p_j = user) \wedge (d = none)$
- Post $(CA_{ij}) \equiv (p_j = root) \wedge (d = none)$
- Post $(HA_{ij}) \equiv (p_j = user) \wedge (d = none)$
- Post $(MiM_{ij}) \equiv (k_j = 1) \wedge (d = none)$
- Post $(CSA_{ij}) \equiv (p_j = root) \wedge (d = high)$
- Post $(BO_{ij}) \equiv (p_j = root) \wedge (d = high)$
- Post$(CDS_{ij}) \equiv (k_j = 1)$
- Post $(DoS_{ij}) \equiv (r_j = 0) \wedge (d = high)$
- Post $(SH_{ij}) \equiv (p_j = root) \wedge (d = high)$
- Post $(PE_{ij}) \equiv (p_j = root) \wedge (d = none)$
- Post $(TwD_{ij}) \equiv (p_j = root) \wedge (d = high)$

15. Initial state:

$(\mathrm{P_{AP}} = \mathrm{root}) \wedge (\forall\, \mathrm{j} \in \{\mathrm{Z, C, F, U}\}\colon (\mathrm{P_j} = \mathrm{none}) \wedge (\mathrm{k_j} = \mathrm{none}) \wedge (\mathrm{r_j} = 1) \wedge (\mathrm{d} = \mathrm{none}))$.

16. Security property (φ): is that the attacker cannot disrupt/compromise the vehicle (the CAN bus always responds to ECUs, attacker has no root privilege on the CAN bus, and the level of danger is none). The property is written by CTL as:

$\varphi \equiv \mathrm{AG}\,((r_C = 1) \wedge (p_C = none) \wedge (d = none)) \equiv \mathrm{AG}\,(\neg((r_C = 0) \wedge (p_C = root) \wedge (d = high)))$.

5.4. VNS Attack Scenarios Generation

Considering the given security property φ in which the attacker aims to disrupt the system by tampering with the data sent and received on the CAN bus, thus affecting its response, the JKind model checker generates the following counter-example: (*CE1:CA_APCP* → *HA_CPTCU* → *BO_TCUHU* → *CDS_HUC* → *SH_CC* → *TwD_CC*) as a spread-sheet. This sheet has six attack instances such that one attack instance can occur at every time step. By encoding this generated counter example CE1 in disjunct with the property φ being checked and repeating the process until the property is true, twenty CEs are discovered, generating the complete attack graph as shown in Figure 7.

In this graph, one of the attack sequences is described as follows. Initially, the attacker has root access over the access point. The attacker then uses an attack named *RH_APCP,* which gives the attacker root access over the user's cell phone. Afterwards, the *HA_CPTCU* attack is carried out. This attack gives the attacker user access over the telematics unit of the infotainment system. Then, *BO_TCUHU* is conducted to give the attacker root access over the head unit. Once the attacker has gained root access over the head unit, he/she can apply *CDS_HUC*. Here, the attacker uses CAN dump tool to dump all the CAN traffic into the attacker's terminal, then a CAN sniffer tool can be used to filter out the CAN messages that are brought up onto the attacker's terminal. This is done by monitoring which messages change and removing the messages that do not change. The attacker then gains knowledge of the messages used for each function through *SH_CC* attack and sends data to the CAN bus, acting as an authenticated user. This can lead to Denial-of-Service *DoS_CC* attack or many more dangerous attacks on the driver of the vehicle.

It should be noted that the CAN bus lacks any cyber-security countermeasures, and it does not have any form of authentication. One of the ways in which the CAN bus can be improved is by adding an IDS between the CAN bus and the ECUs to detect and alert the owner and the vehicle manufacturers of unauthorized entry to the CAN bus.

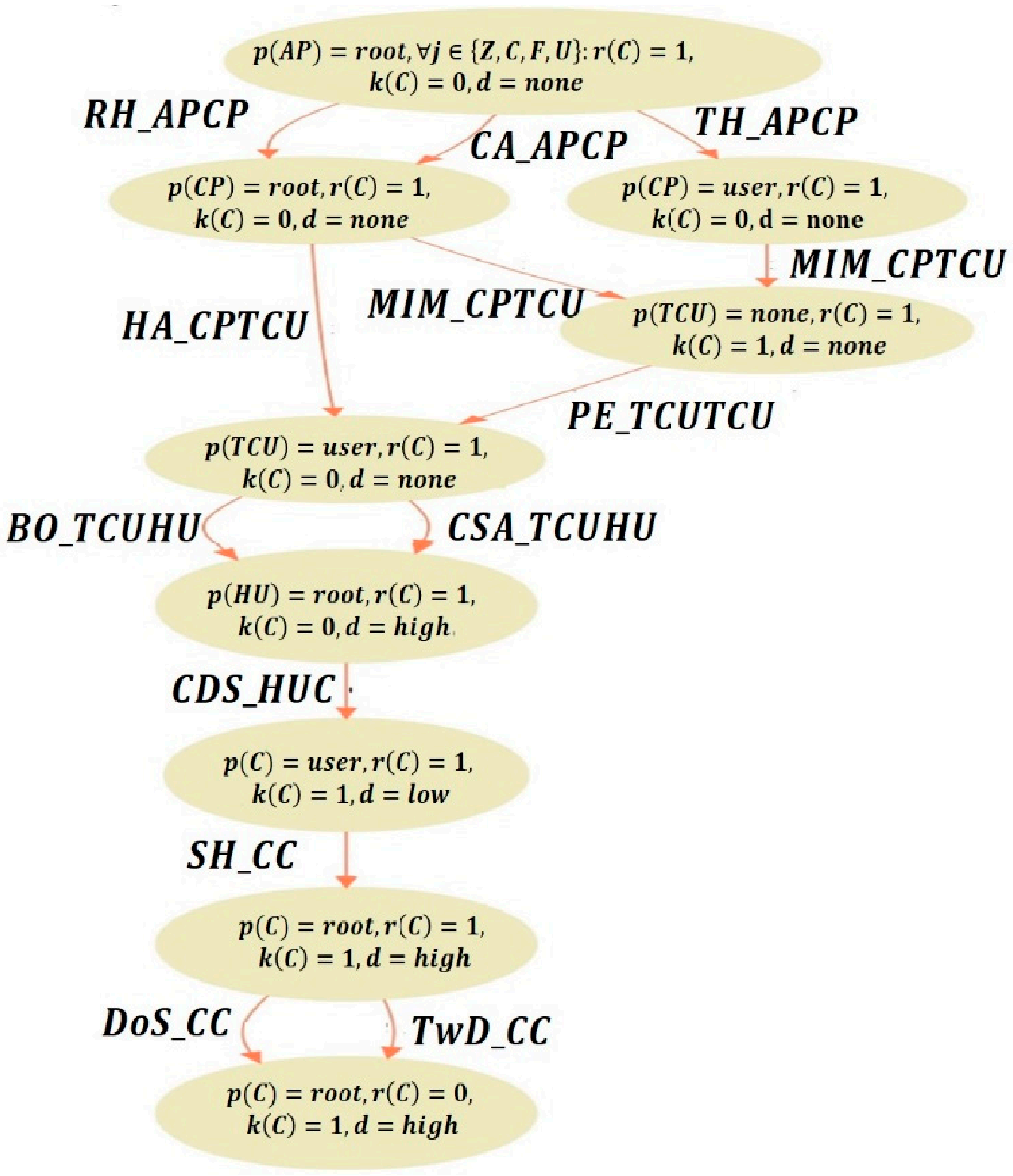

$p(AP) = root, \forall j \in \{Z, C, F, U\}: r(C) = 1, k(C) = 0, d = none$
RH_APCP
CA_APCP
TH_APCP
$p(CP) = root, r(C) = 1, k(C) = 0, d = none$
$p(CP) = user, r(C) = 1, k(C) = 0, d = none$
MIM_CPTCU
HA_CPTCU
MIM_CPTCU
$p(TCU) = none, r(C) = 1, k(C) = 1, d = none$
PE_TCUTCU
$p(TCU) = user, r(C) = 1, k(C) = 0, d = none$
BO_TCUHU
CSA_TCUHU
$p(HU) = root, r(C) = 1, k(C) = 0, d = high$
CDS_HUC
$p(C) = user, r(C) = 1, k(C) = 1, d = low$
SH_CC
$p(C) = root, r(C) = 1, k(C) = 1, d = high$
DoS_CC
TwD_CC
$p(C) = root, r(C) = 0, k(C) = 1, d = high$

Figure 7. Vehicular network system (VNS) attack graph.

Overall, while our scheme does not illuminate the attack, it graphically shows the potential sequences of attack instances an attacker can seek to disrupt the system. In fact, the generated graphs may aid system administrators to decide the placement of appropriate detection and prevention measures. For instance, for the three case studies—NPP, ICS, and VNS—experimental results show that the common attack among their attack graphs is DoS. Thus, identifying such an attack using, for example, IDS, may render the violation of the security properties. Also, for NPP, the generated attack graph illustrates that a DoS attack can never be conducted correctly without running an MI attack first. Thus, by way of preventing the MI attacks, the system administrators can also eliminate the DoS attacks, which would significantly enhance the system security.

For ICS, the generated attack graph requires as an initial step the running of either SE or P attacks against the enterprise site management computer. This can be done by exploiting the COTS vulnerability in the operating system of this computer. Therefore, securing the operating system may prevent the attacker from executing the remaining attacks. In regard to VNS, the generated attack graph demonstrates that the graph cannot be generated unless one of the three attacks (RH, CA, and TH) is executed initially against the driver's cell phone, thus securing the driver's cell phone would prevent such attacks.

It can be seen that the number of pre and post conditions is linear in the number of attack instances and the dynamic state variables [44]. In addition, the complexity is known to be polynomial in the size of the model and the length of the security property [63]. The accuracy of the proposed scheme depends on how accurate the system architecture model is formally specified. Similar to our scheme, the scheme proposed by [17] requires having architectural information of the system model. Thus, the implemented attacks are more detailed and can capture more information about the potential attacker actions. In addition, in [17], the system and the attacker are modeled with behavioral rules using graph transformation system. However, the generated state space (attack graph) is quite complex and large, thus requiring attack graph to attack tree transformation, while in our scheme, such behavioral rules are captured through pre and post conditions of attacker actions in the AADL model. The developed visualizer Windows application enhances the efficiency and the ease of use for the underlying long spreadsheets attack scenarios into an appealing visual attack graph. The shortcoming of our scheme is that it requires accessibility of the system model. This requires a specific and accurate one-time modeling exertion to obtain the system characterization for elements, connectivity, assets, and their vulnerabilities.

6. Conclusions

In this paper, we illustrated a model-based attack graph implementation and its graphical visualization for three CPSs case studies—an NPP system, an ICS, and a VNS—using JKind model checker and Microsoft Visual Studio integrated development environment (IDE). The Visual Studio program can read all scenarios spreadsheets and automatically visualize the potential attack sequences, their final state post-conditions, and CEs reduced spreadsheet viewer. The main criterion for modeling is the application of an architectural specification language to obtain the security-related information of the system. The generated attack graph can benefit system administrators to select the best arrangement of countermeasures, preventing the occurrence of such attacks in addition to cyber security risk assessment. For future work, we aim to enhance the GUI to automatically present the associated resilience levels of CPS, thus resulting in a hybrid attack graph.

Author Contributions: Conceptualization, M.I.; Methodology, M.I.; Software, M.I., Q.A.-H., R.E., A.A., O.A.; Validation, M.I., Q.A.-H.; Formal Analysis M.I.; Investigation, M.I., Q.A.-H., R.E., A.A.; Resources, M.I.; Data Curation, M.I., Q.A.-H., R.E.; Writing—Original Draft Preparation, M.I., Q.A.-H., R.E., O.A.; Writing—Review & Editing, M.I.; Visualization, M.I., Q.A.-H., R.E.; Supervision, M.I.; Project Administration, M.I.; Funding Acquisition, M.I. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Deanship of Graduation Studies and Scientific Research at the German Jordanian University for the seed fund SATS 02/2018.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shah, S.; Ren, A.; Fan, D.; Zhang, Z.; Zhao, N.; Yang, X.; Luo, M.; Wang, W.; Hu, F.; Rehman, M.; et al. Internet of things for sensing: A case study in the healthcare system. *Appl. Sci.* **2018**, *8*, 508. [CrossRef]
2. Chen, C.M.; Xiang, B.; Wu, T.Y.; Wang, K.H. An anonymous mutual authenticated key agreement scheme for wearable sensors in wireless body area networks. *Appl. Sci.* **2018**, *8*, 1074. [CrossRef]
3. Lee, J.; Kim, S. Emergency-prioritized asymmetric protocol for improving QoS of energy-constraint wearable device in wireless body area networks. *Appl. Sci.* **2018**, *8*, 92. [CrossRef]
4. Zikria, Y.B.; Kim, S.W.; Hahm, O.; Afzal, M.K.; Aalsalem, M.Y. Internet of Things (IoT) Operating Systems Management: Opportunities, Challenges, and Solution. *Sensors* **2019**, *19*, 1793. [CrossRef] [PubMed]

5. Al-Mohannadi, H.; Mirza, Q.; Namanya, A.; Awan, I.; Cullen, A.; Disso, J. Cyber-attack modeling analysis techniques: An overview. In Proceedings of the 4th International Conference on Future Internet of Things and Cloud Workshops, Vienna, Austria, 22–24 August 2016; pp. 69–76.
6. Yuan, B.T.; Pan, Z.L.; Fan, S.H. A Review on Network Attack Graph Technology. In Proceedings of the International Conference on Electrical, Control, Automation and Robotics (ECAR 2018), Xiamen, China, 16–17 September 2018; pp. 239–245.
7. Carnegie-Mellon-University. Open Source Aadl Tool Environment for the SAE Architecture. 2018. Available online: http://osate.github.io/index.html (accessed on 15 May 2018).
8. Sheeran, M.; Singh, S.; Stålmarck, G. Checking Safety Properties Using Induction and a SAT-Solver. In Proceedings of the International Conference on Formal Methods in Computer-Aided Design, Austin, TX, USA, 1–3 November 2000.
9. Coorporation, M. Visual Studio. 2018. Available online: https://visualstudio.com/vs/ (accessed on 15 May 2018).
10. Ibrahim, M.; Al-Hindawi, Q. Attack Graph Modeling for Nuclear Power Plant. In Proceedings of the 10th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Iasi, Romania, 28–30 June 2018.
11. Olifer, D.; Goranin, N.; Cenys, A.; Kaceniauskas, A.; Janulevicius, J. Defining the Minimum Security Baseline in a Multiple Security Standards Environment by Graph Theory Techniques. *Appl. Sci.* **2019**, *9*, 681. [CrossRef]
12. Kaynar, K.; Sivrikaya, F. Distributed attack graph generation. In *IEEE Transactions on Dependable and Secure Computing*; IEEE Computer Society Press: Los Alamitos, CA, USA, 2015; Volume 13, pp. 519–532.
13. Johnson, P.; Vernotte, A.; Ekstedt, M.; Lagerström, R. pwnpr3d: An attack-graph-driven probabilistic threat-modeling approach. In Proceedings of the 11th IEEE International Conference on Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 278–283.
14. Kaynar, K. A taxonomy for attack graph generation and usage in network security. *J. Inf. Secur. Appl.* **2016**, *29*, 27–56. [CrossRef]
15. Lallie, H.S.; Debattista, K.; Bal, J. An empirical evaluation of the effectiveness of attack graphs and fault trees in cyber-attack perception. *IEEE Trans. Inf. Forensics Secur.* **2017**, *13*, 1110–1122. [CrossRef]
16. Taylor, C.R.; Venkatasubramanian, K.; Shue, C.A. Understanding the security of interoperable medical devices using attack graphs. In Proceedings of the 3rd International Conference on High Confidence Networked Systems, Berlin, Germany, 15–17 April 2014; pp. 31–40.
17. Karray, K.; Danger, J.L.; Guilley, S.; Elaabid, M.A. Attack Tree Construction and Its Application to the Connected Vehicle. In *Cyber-Physical Systems Security*; Springer: Cham, Switzerland, 2018; pp. 175–190.
18. Fredj, O.B. A realistic graph-based alert correlation system. *Secur. Commun. Netw.* **2015**, *8*, 2477–2493. [CrossRef]
19. Luckett, P.; McDonald, J.T.; Glisson, W.B. Attack-graph threat modeling assessment of ambulatory medical devices. In Proceedings of the 50th Hawaii International Conference on System Sciences, Waikoloa Village, HI, USA, 4–7 January 2017; pp. 3648–3657.
20. Stan, O.; Bitton, R.; Ezrets, M.; Dadon, M.; Inokuchi, M.; Ohta, Y.; Yamada, Y.; Yagyu, T.; Elovici, Y.; Shabtai, A. Extending Attack Graphs to Represent Cyber-Attacks in Communication Protocols and Modern IT Networks. *arXiv* **2019**, arXiv:1906.09786.
21. Pei, K.; Gu, Z.; Saltaformaggio, B.; Ma, S.; Wang, F.; Zhang, Z.; Si, L.; Zhang, X.; Xu, D. Hercule: Attack story reconstruction via community discovery on correlated log graph. In Proceedings of the 32nd Annual Conference on Computer Security Applications, Los Angeles, CA, USA, 5–9 December 2016; pp. 583–595.
22. Jia, F.; Hong, J.B.; Kim, D.S. Towards automated generation and visualization of hierarchical attack representation models. In Proceedings of the IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool, UK, 26–28 October 2015; pp. 1689–1696.
23. Zhang, Y.; Wang, L.; Xiang, Y.; Ten, C.W. Power system reliability evaluation with SCADA cybersecurity considerations. *IEEE Trans. Smart Grid* **2015**, *6*, 1707–1721. [CrossRef]
24. Kure, H.; Islam, S.; Razzaque, M. An integrated cyber security risk management approach for a cyber-physical system. *Appl. Sci.* **2018**, *8*, 898. [CrossRef]

25. Agmon, N.; Shabtai, A.; Puzis, R. Deployment optimization of IoT devices through attack graph analysis. In Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, Miami, FL, USA, 15–17 May 2019; pp. 192–202.
26. Ge, M.; Kim, D.S. A framework for modeling and assessing security of the internet of things. In Proceedings of the IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), Melbourne, Australia, 14–17 December 2015; pp. 776–781.
27. Singhal, A.; Ou, X. Security risk analysis of enterprise networks using probabilistic attack graphs. In *Network Security Metrics*; Springer: Cham, Switzerland, 2017; pp. 53–73.
28. Noel, S.; Jajodia, S. A suite of metrics for network attack graph analytics. In *Network Security Metrics*; Springer: Cham, Switzerland, 2017; pp. 141–176.
29. Ekelhart, A.; Kiesling, E.; Grill, B.; Strauss, C.; Stummer, C. Integrating attacker behavior in IT security analysis: A discrete-event simulation approach. *Inf. Technol. Manag.* **2015**, *16*, 221–233. [CrossRef]
30. Durkota, K.; Lisý, V.; Bošanský, B.; Kiekintveld, C. Optimal network security hardening using attack graph games. In Proceedings of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015; pp. 526–532.
31. Hong, J.B.; Kim, D.S.; Chung, C.J.; Huang, D. A survey on the usability and practical applications of graphical security models. *Comput. Sci. Rev.* **2017**, 1–16. [CrossRef]
32. Bozzelli, L.; Molinari, A.; Montanari, A.; Peron, A.; Sala, P. Interval vs. point temporal logic model checking: An expressiveness comparison. *ACM Trans. Comput. Log.* **2018**, *20*, 4–41. [CrossRef]
33. Hliněný, P.; Pokrývka, F.; Roy, B. FO model checking on geometric graphs. *Comput. Geom.* **2019**, *78*, 1–19. [CrossRef]
34. Wang, M.; Liu, C.; Xie, T.; Sun, Z. Data-driven model checking for errors-in-variables varying-coefficient models with replicate measurements. *Comput. Stat. Data Anal.* **2020**, *141*, 12–27. [CrossRef]
35. Deng, M.; Cao, H.; Zhu, W.; Wu, H.; Zhou, Y. Benchmark Tests for the Model-Checking-Based IDS Algorithms. *IEEE Access* **2019**, *7*, 135479–135498. [CrossRef]
36. Wang, H.; Zhong, D.; Zhao, T.; Ren, F. Integrating Model Checking With SysML in Complex System Safety Analysis. *IEEE Access* **2019**, *7*, 16561–16571. [CrossRef]
37. Mihalache, S.F.; Pricop, E.; Fattahi, J. Resilience Enhancement of Cyber-Physical Systems: A Review. In *Power Systems Resilience*; Springer: Cham, Switzerland, 2019; pp. 269–287.
38. Jančík, P.; Kofroň, J.; Alt, L.; Fedyukovich, G.; Hyvärinen, A.E.; Sharygina, N. Exploiting partial variable assignment in interpolation-based model checking. In *Formal Methods in System Design*; Springer: Cham, Switzerland, 2019; pp. 1–39.
39. Jensen, L.S.; Kaufmann, I.; Larsen, K.G.; Nielsen, S.M.; Srba, J. Model checking and synthesis for branching multi-weighted logics. *J. Log. Algebraic Methods Program.* **2019**, *105*, 28–46. [CrossRef]
40. Varuttamaseni, A.; Bari, R.A.; Youngbl, R. Construction of a Cyber Attack Model for Nuclear Power Plants. In Proceedings of the ANS Annual Conference, San Francisco, CA, USA, 11–15 June 2017.
41. Martin, R.A. Managing vulnerabilities in your commercial-off-the-shelf (COTS) systems using an industry standards effort. In Proceedings of the 21st Digital Avionics Systems Conference, Irvine, CA, USA, 27 October 2002; p. 4A1.
42. Basnight, Z.; Butts, J.; Lopez, J. Firmware modification attacks on programmable logic controllers. *Int. J. Crit. Infrastruct. Prot.* **2013**, *6*, 76–84. [CrossRef]
43. Abdul Razak, T.; Ibrahim Salim, M. A study on IDS for preventing Denial of Service attack using outliers techniques. In Proceedings of the 2nd IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, India, 17–18 March 2016.
44. Al Ghazo, A.; Ibrahim, M.; Ren, H.; Kumar, R. A2G2V: Automatic Attack Graph Generation and Visualization and Its Applications to Computer and SCADA Networks. *IEEE Trans. Syst. Man Cybern Syst.* **2019**, 1–11, (Early Access). [CrossRef]
45. Gacek, A.; Backes, J.; Whalen, M.; Wagner, L.; Ghassabani, E. The JKind Model Checker. In Proceedings of the Computer Aided Verification 2018, Oxford, UK, 14–17 July 2018.
46. David, M. Visual Studio IDE Offers Many Advantages for Developers. SearchSoftware Quality. Available online: https://searchsoftwarequality.techtarget.com (accessed on 26 May 2018).
47. JKind, An Infinite-State Model Checker for Safety Properties. Loonwerks. Available online: http://loonwerks.com/tools/jkind.html (accessed on 26 May 2018).

48. Halbwachs, N.; Caspi, P.; Raymond, P.; Pilaud, D. The synchronous data flow programming language LUSTRE. *Proc. IEEE* **1991**, *79*, 1305–1320. [CrossRef]
49. The Assume Guarantee Reasoning Environment. Loonwek. Available online: http://loonwerks.com/tools/agree.html (accessed on 27 May 2018).
50. Sklyar, V. Cyber Security of Safety-Critical Infrastructures: A Case Study for Nuclear Facilities. Information & Security. *Int. J.* **2012**, *28*, 98–107.
51. Zulkurnain, A.U.; Kamal, A.; Kamarun, B. Social Engineering Attack Mitigation. *Int. J. Math. Comput. Sci.* **2015**, *1*, 188–198.
52. Meixell, B.; Forner, E. Out of Control: Demonstrating SCADA Exploitation. In Proceedings of the Black Hat 2013 Conference, Las Vegas, LV, USA, 27 July–1 August 2013.
53. Sayegh, N.; Chehab, A.; Elhajj, I.H. Internal security attacks on SCADA systems. In Proceedings of the 2013 3rd International Conference in Communication and Information Technology (ICCIT), Beirut, Lebanon, 19–21 June 2013; pp. 22–27.
54. Papp, D.; Ma, Z.; Buttyan, L. Embedded systems security: Threats, vulnerabilities, and attack taxonomy. In Proceedings of the 2015 13th Annual Conference on Privacy, Secur Trust (PST), Izmir, Turkey, 21–23 July 2015; pp. 145–152.
55. Scarfone, K.; Mell, P. *Guide to Intrusion Detection and Prevention Systems (IDPS)*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007; pp. 1–127.
56. Hartzell, S.; Stubel, C. Automobile CAN Bus Network Security and Vulnerabilities (2018). Available online: https://canvas.uw.edu/files/47669787/download?download_frd=1 (accessed on 15 July 2019).
57. Asvestopoulos, A. Intrusion Protection of in-Vehicle Network: Study and Recommendations. Available online: www.kth.se (accessed on 27 May 2019).
58. Jamie, riden. CLIENT-SIDE ATTACKS. 2008. Available online: https://www.sciencedirect.com/topics/computer-science/client-side-attack7 (accessed on 28 July 2019).
59. Khandelwal, S. New Drammer Android Hack Lets Apps Take Full Control (root) of Your Phone. 2016. Available online: https://thehackernews.com/2016/10/root-android-phone-exploit.html (accessed on 15 July 2019).
60. Evenchick, E. Hopping on the CAN bus. In Proceedings of the Black Hat Asia, Marina Bay Sands, Singapore, 24–27 March 2015.
61. CAN SNIFFER-Technical Description, CANLAB s.r.o. Available online: http://www.canlab.cz. (accessed on 15 July 2019).
62. Kushawah, S. Most Popular Types of WiFi Cyberattacks. Available online: https://socialwifi.com/knowledge-base/network-security/most-popular-types-wifi-cyberattacks/ (accessed on 15 July 2019).
63. Schnoebelen, P. The Complexity of Temporal Logic Model Checking. *Adv. Modal Log.* **2002**, *4*, 35–79.