Attack models for industrial control system elements based on a graph approach and countermeasures

Oleksii Novikov[1,†], Iryna Stopochkina[1,*,†], Andrii Voitsekhovskyi[1,†] and Mykola Ilin[1,†]

[1] Igor Sikorsky Kyiv Polytechnic Institute, Beresteiskyi Ave 37, 03056 Kyiv, Ukraine

Abstract

This study investigates cyberattacks targeting industrial control systems (ICS) and proposes a formal methodology for their analysis. Typical attack progression scenarios are identified and modeled using logical attack graphs. These graphs are transformed into Boolean expressions and then into Conjunctive Normal Form (CNF) to enable automated analysis via SAT/SMT solvers.

The proposed methodology allows for determining whether a given attack is feasible within a specific system configuration, identifying minimal configurations that either ensure security or leave the system vulnerable. It also considers the presence of irremediable vulnerabilities and incorporates system usability constraints into the analysis process. To demonstrate the approach, a computational experiment was conducted using the z3-python library, highlighting the applicability of the method in evaluating real-world ICS security scenarios.

The proposed approach can assist in designing and implementing effective countermeasures for protecting critical infrastructure systems from cyber threats.

Keywords

cybersecurity, industrial control systems, attack graph, embedded systems, SAT/SMT

1. Introduction

Industrial control systems (ICS) are frequently targeted by various types of cyberattacks [1]. Enhancing mathematical tools for the analysis of system security, attack prerequisites, attack progression, and potential consequences remains a pressing challenge. Attacks on industrial control systems typically focus on standard components, for which emulators and testbeds have been developed to facilitate research [2], [3]. Analytical tools used for studying such attacks include not only practical experimentation but also formal models, such as state-space representations [2], [4], and graph-based attack models [5]. These models offer several advantages, including the ability to recognize recurring patterns in complex attacks, leverage graph-specific algorithms, and store data in graph databases for further analysis and relationship extraction.

One of the challenges in this domain is the construction of attack graphs for large-scale networks. As demonstrated in [6], attack graphs in real-world systems can comprise hundreds of nodes. This necessitates the use of security scanners capable of identifying system vulnerabilities. Several tools exist that can analyze a system and construct a logical representation of an attack graph [7]–[9]. The structure and underlying logic of such tools are examined in detail in [10].

On the other hand, logical graph-based models can be interpreted as Boolean expressions [11], [12], which can be further analyzed using specialized tools [13]. Among these tools are SAT (Boolean Satisfiability) and SMT (Satisfiability Modulo Theories) solvers, which are designed to efficiently

ITS-2024: Information Technologies and Security, December 19, 2024, Kyiv, Ukraine

[*] Corresponding author.

[†] These authors contributed equally.

o.novikov@kpi.ua (O. Novikov); i.stopochkina@kpi.ua (I. Stopochkina); a.voitsekhovskyi@kpi.ua (A. Voitsekhovskyi); m.ilin@kpi.ua (M. Ilin)

0000-0001-5988-3352 (O. Novikov); 0000-0002-0346-0390 (I. Stopochkina); 0009-0004-6009-9492 (A. Voitsekhovskyi); 0000-0002-1065-6500 (M. Ilin)

solve computationally complex problems [14], [15]. In the case of large attack graphs characterized by a significant number of states, the corresponding Boolean expressions may involve a vast number of variables. As a result, exhaustive enumeration of all possible configurations to find optimal ones becomes computationally infeasible within a reasonable timeframe. Therefore, it is essential to employ specialized verification tools such as SAT/SMT solvers. Using SAT/SMT solvers allows for the verification of formula satisfiability under specific constraints (e.g., usability and security policies) and enables the identification of variable assignments that satisfy the formula – effectively revealing the privileges and configurations under which an attack becomes feasible. Additionally, the solver can be used to determine minimal configurations and privilege distributions that prevent an attacker from successfully executing an attack.

In this study, we propose logical graph-based models for several types of typical attacks, the feasibility of which was validated in a controlled laboratory environment. Preventive measures against such cyber-physical attacks are also proposed. The corresponding logical attack graphs were converted into Boolean expressions, and scripts were developed using the z3-python SMT solver [16]. These scripts enabled the analysis of system configurations and privilege distributions that either permit or prevent the realization of the modeled attacks. This integration of graph-based modeling with formal verification techniques provides a novel and practical methodology for security analysis and risk mitigation in ICS environments.

2. Cyber attacks analysis

To analyze the stages of typical cyberattacks, we use ICS laboratory, assembled as part of a project supported by USAID [17], which includes technological testbeds, which are built using standard hardware and software components commonly employed in industrial control systems. The technologies, communication protocols, and software-hardware tools used in these testbeds are designed to address a wide range of tasks within critical infrastructure environments. Laboratory testbeds include: ventilation and cooling system tubing with control devices; water tanks integrated with control system components; production line testbed equipped with Human-Machine Interface (HMI), production line platform, and IFM Safety system used for regulating the platform's behavior in critical modes.

A structured summary of the examined attacks is presented in Table 1.
These include:

1. Direct packet injection using the S7COMM protocol [18]. By utilizing the Python library snap7 [19], it was possible to simulate the transmission of control commands to PLCs without authentication. For example, in the water level control system, the operational settings of the pump were altered.
2. Man-in-the-middle (MITM) attack via ARP poisoning, performed using the Ettercap tool. This attack enabled interception of all traffic between the operator and the controller, allowing data manipulation at the attacker's discretion. In the cooling system, for instance, the fan speed could be increased to its maximum, potentially causing mechanical damage.
3. Remote Code Execution (RCE) on IFM controllers by exploiting an open Telnet port [20] with default credentials (target:target). This vulnerability allowed commands to be executed directly on the controller without interacting with the operator's programming environment.
4. Supply Chain attack involving the insertion of a backdoor into the firmware. This attack scenario was examined in the context of a production line system.
5. Privilege Escalation, in which a vulnerability in the IFM controller kernel enabled the acquisition of superuser privileges.

MITM attack description. Initial Setup:

1. In the example of the laboratory training testbeds, the available local network infrastructure is illustrated in Figures 1-3.

Table 1
Attacks on Industrial Control System Components

Attack Type	Target	Access to Facility Network	Operator Panel Access	Vulnerability description	Target system
Direct Packet Injection	Control System	+	-	Unprotected S7COMM protocol, lack of authentication on the device.	Water level control system, cooling system
Data Substitution (MITM)	Control System	+	-	Unprotected S7COMM protocol, lack of encryption.	Cooling system
Supply Chain Attack	Controller	-	-	Firmware supply chain vulnerability, potential for malicious code injection.	IFM
Remote Code Execution	Control System	+	-	Open Telnet port with default credentials (target:target).	IFM
Privilege Escalation	Controller	+	-	Vulnerable controller kernel, possibility of executing kernel-level exploit to gain root access.	IFM

2. The Control Center is a Windows 11 machine equipped with a communication panel for interacting with physical equipment, as well as a monitoring and control system for observing equipment parameters.
3. The PLC (Programmable Logic Controller) contains firmware that directly controls specific physical processes on the equipment and communicates with the Control Center over the network to receive commands and transmit operational data.
4. Additional Windows 11 machines in the network perform various user-level functions.

The system scenario corresponds to a configuration implemented in the laboratory environment. The Control Center and the PLC communicate via the S7Comm protocol. The physical equipment in this case includes air compressor, pump. The PLC maintains a database containing the current state of the equipment (e.g., pressure in a pipe, fluid level in a water tank, pump power, and the rules governing its adjustment). Operation of the physical devices is carried out through firmware functions on the PLC, which retrieve input arguments from this database.

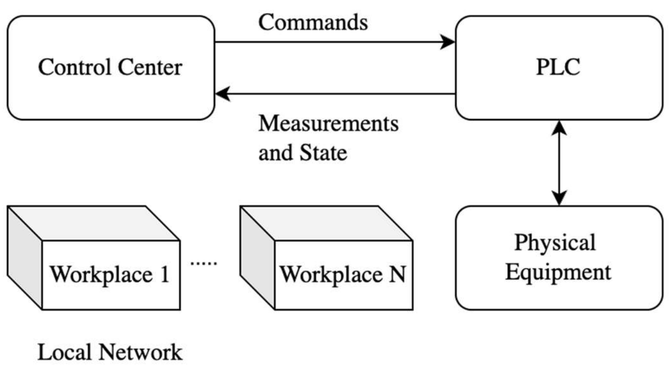


Figure 1: General structure of the laboratory network used for ICS cybersecurity experiments.

The Control Center can read from and write to this database using the S7Comm protocol. This enables interaction between the operator's graphical user interface (GUI) and the physical equipment. The original Control Center and PLC are software-protected, and the attacker does not have direct access to them – only network-level access.

The attacker's scenario involves compromising a machine within the local network (Figure 1). In this setup, strict network policies (e.g., firewall rules) are either absent or have already been bypassed.

The objective of the attack is to gain control over the physical equipment without causing any noticeable changes in the graphical displays of the operator's interface. The sequential steps of the attack are illustrated in Figures 2 and 3.

The attacker performs a Man-in-the-Middle (MITM) attack within the local network, placing themselves between the Control Center and the PLC (Figure 2). An ARP spoofing technique is used to impersonate each device to the other, enabling the interception and manipulation of all traffic exchanged between them. The attacker gains the ability to analyze the specifics of the communication protocol and to understand how internal communication between the Control Center and the PLC operates under the S7Comm protocol. As a result, the attacker identifies which memory cells in the PLC's internal database correspond to specific system parameters, as well as the functions transmitted by the Control Center.

The attacker either records a legitimate response packet from the PLC to a request issued by the Control Center or crafts a custom response packet based on the observed structure.

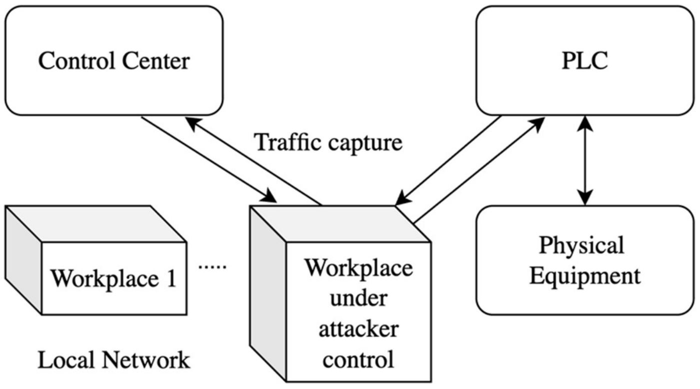


Figure 2: Network traffic capture during a Man-in-the-Middle (MITM) attack.

After analyzing intercepted traffic, the attacker crafts a malicious interface that mimics the legitimate Control Center, enabling unauthorized command injection and data spoofing without detection by the operator (Figure 3).

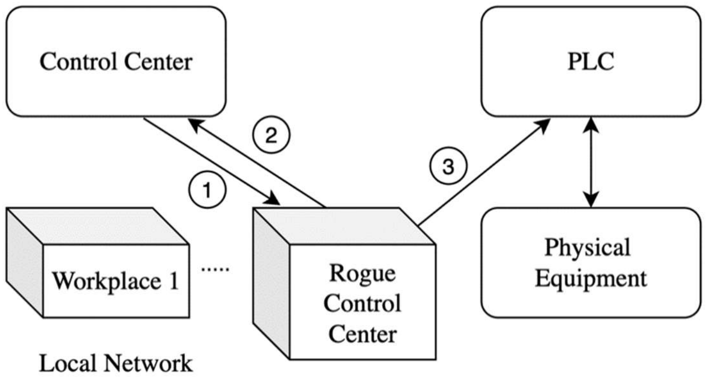


Figure 3: Creation of a rogue Control Center by the attacker.

The attacker then reinitiates an active MITM attack (Figure 2). However, this time, any request issued by the Control Center is intercepted, and the attacker returns falsified status information – such as replaying a previously recorded valid state or injecting spoofed values that mask the actual behavior of the physical system.

IFM attack description. Another type of attack may be realized as a result of vulnerabilities [21], [22] present in certain embedded systems, such as the IFM Safety PLC, which receives signals from physical equipment and controls its behavior in the event of faults. Manufacturers frequently embed hardcoded credentials into the firmware of such devices to facilitate technical support, ease of deployment during production, legacy protocol compatibility, or other reasons. These credentials are typically immutable – logins are often identical across all devices, and passwords are either weak or follow predictable formats. Once network access to the device is gained, these credentials may be exploited by an attacker.

An example of such an attack is demonstrated for IFM security system, in a scenario where the firmware includes an active Telnet service running with root privileges. The attacker can use the hardcoded credentials to establish a Telnet session and gain elevated access to the system.

Upon obtaining administrative privileges, the attacker may modify the safety logic and overwrite the controller's control program. For instance, this may involve disabling the emergency shutdown response, triggering false emergency reactions during normal operation, or disabling the processing of signals from visual indicators or motion sensors. The attacker may also disable relays responsible for stopping mechanical systems. In addition, the controller can be compromised by implanting malicious code, such as creating a hidden user account to maintain unauthorized access, or deploying a script that is triggered by a specific sensor reading or event.

Remarks: A known kernel-level vulnerability in the controller [23] – related to a race condition – enables local privilege escalation. Exploiting this vulnerability requires address space probing on the device. However, this is not always feasible, as the device has limited computational resources, and brute-force attempts may require several days. To mitigate such attacks, it is recommended to use cryptographically secured communication protocols (e.g., S7Comm+), wherever possible. Additionally, the implementation of appropriate traffic monitoring and filtering policies (e.g., SIEM (Security Information and Event Management) systems and network firewalls) within the local network is critical. As further countermeasures, firmware should be updated to patched versions that eliminate known vulnerabilities, remote access should be restricted or disabled entirely, and external authentication mechanisms should be employed to reinforce access control.

Attack using CaddyWiper malware description

To illustrate more sophisticated attacks that operate across multiple layers of the information and operational systems within a critical infrastructure environment, we examine a complex attack involving the CaddyWiper malware [24], which was specifically designed to target the Ukrainian energy sector [25].

A schematic representation of this attack across the layers of the Purdue Model for ICS Security is shown in Figure 4.

The attack sequence can be outlined with reference to the techniques classified under the MITRE ATT&CK framework.

The diagram (Figure 4) illustrates how the attack progresses from initial access (e.g., phishing, remote services) through lateral movement, malware execution, and impact, affecting both IT (Information Technology) and OT (Operation Technology) layers. Each stage corresponds to specific tactics and techniques as defined in the MITRE ICS matrix, highlighting the multi-layered nature of advanced persistent threats targeting critical infrastructure.

The Initial Access phase is achieved using a variety of techniques, given in [26]. For instance, adversaries may deploy CaddyWiper onto corporate devices with access to SCADA (Supervisory Control and Data Acquisition) systems via technique marked as T0865 (Spearphishing Attachment) [27], by sending phishing emails containing malicious documents. Initial access may also be obtained through T0822 (External Remote Services) [28], for example, by exploiting exposed RDP or VPN services using stolen credentials to gain access to SCADA engineering workstations. In cases where insufficient attention is paid to supply chain security, T0862 (Supply Chain Compromise) [29] may be employed, whereby SCADA software updates or third-party contractors become vectors for malware injection.

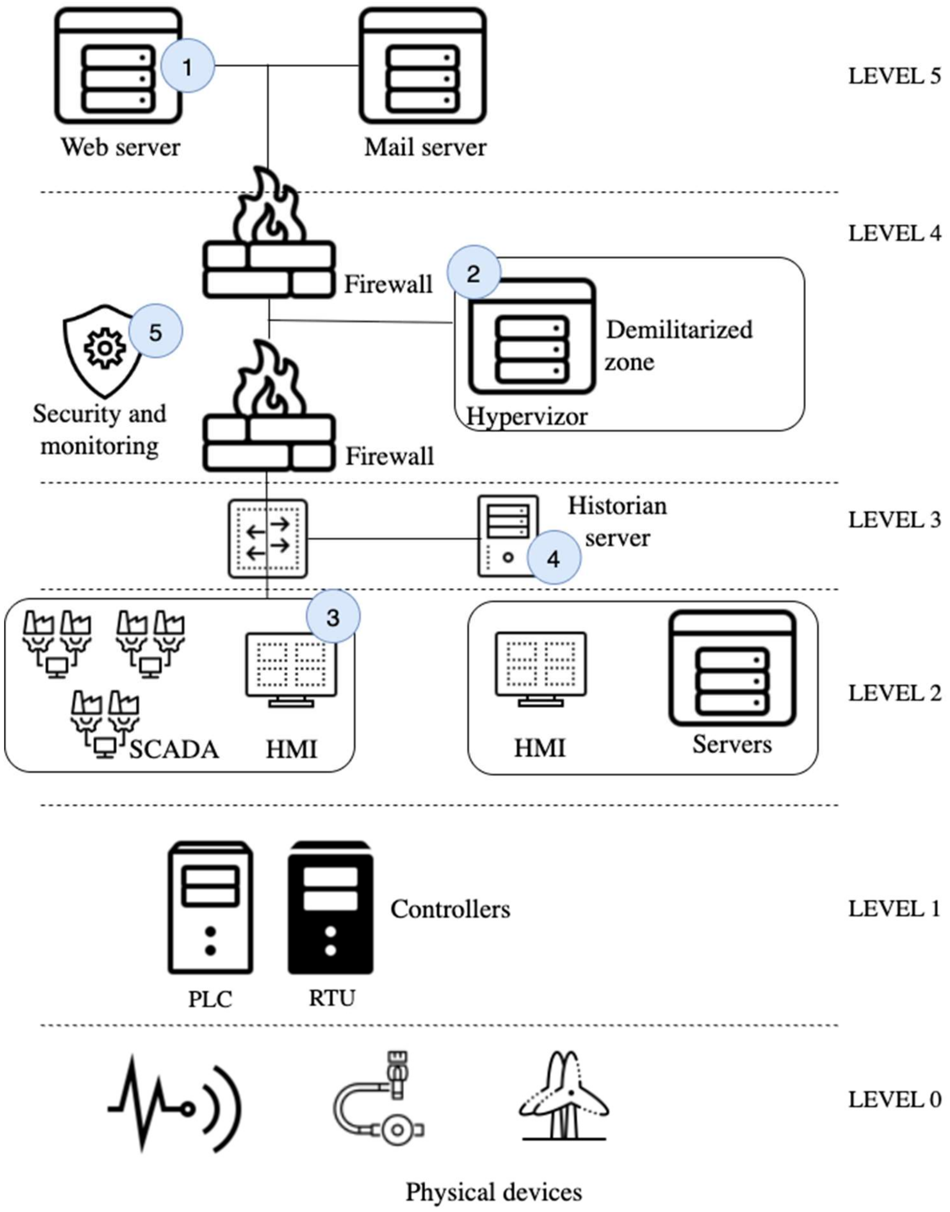


Figure 4: Lifecycle of the CaddyWiper malware attack. Marks 1-5 indicate attack steps.

The next stage involves exploitation of vulnerabilities and privilege escalation. This may be performed using T0819 (Exploit Public-Facing Application) [30], which targets vulnerabilities in SCADA systems or web interfaces of PLC controllers. Alternatively, T0859 (Valid Accounts) [31] may be used to steal or substitute credentials in order to access SCADA servers. Adversaries may extend access laterally through T0847 [32] or other techniques of lateral movement, allowing CaddyWiper to propagate across the network from an initially compromised machine using mechanisms such as PsExec, Windows Management Instrumentation (WMI), or flaws in Active Directory Group Policy configurations.

During the Execution phase on SCADA systems, adversaries may employ T0853 (Scripting) [33] by utilizing PowerShell or VBScript to deliver and execute CaddyWiper. Another potential technique is T0857 (System Firmware) [34], involving overwriting of the Master Boot Record (MBR) to disable SCADA components.

The Impact phase involves the actual disruption of SCADA functionality using T0809 (Data

Destruction) [35]. This results in the deletion of SCADA configuration files and operational data, potentially leading to data loss and shutdown of industrial processes. The malware may also destroy backups and wipe log files to hinder recovery efforts. Furthermore, T0813 (Denial of Control) [36] may be executed to sever SCADA components from control over field devices (e.g., pumps, turbines, generators).

The final phase focuses on anti-forensics and cover-up. This corresponds to T0872 (Indicator Removal on Host) [37], wherein CaddyWiper deletes itself after execution to avoid post-incident analysis. Additionally, T0809 (Data Destruction) is used to erase Historian server data, resulting in the loss of SCADA process history and depriving operators of critical event information.

The identification of vulnerabilities and insecure configurations that may be exploited by an attacker can be conducted in a semi-automated manner using network security scanning tools.

3. Attack models in form of graphs and Boolean expressions

3.1. Graph models and Boolean representation of examined cyberattacks

In this chapter, we present models of cyberattacks in the form of logical attack graphs. A logical attack graph is defined as $G = \{A, P, C; E\}$, where A, P, and C are sets of vertices, and E is the set of edges. The vertices $a_i \in A$ represent direct exploits or attack actions. These nodes logically link the prerequisites (system configurations) to the consequences (privileges $p_i \in P$) that the attacker may acquire.

The vertices $c_i \in C$ denote system configurations, which may either reflect implemented cybersecurity measures or inherent system vulnerabilities. Graphically, the elements are represented as follows: a_i (attacks) as ovals, p_i (privileges) as diamonds, and c_i (configurations) as rectangles.

When multiple edges lead into an attack vertex a_i, the vertex functions as a logical AND (conjunction), indicating that all incoming conditions must be satisfied to enable the attack. In contrast, when multiple edges lead into a privilege vertex p_i, it behaves as a logical OR (disjunction), meaning that the privilege can be gained if any of the conditions are met.

Privileges p_i may be acquired independently or as a result of exploiting vulnerable configurations. The attack vertices a_i themselves are not explicitly included in the Boolean expressions, as they are considered intermediate steps resulting from prior conditions.

The overall outcome of an attack scenario is represented by a Boolean value F_i:

- $F_i = True$ indicates that the attack is feasible under the given configurations and privileges.
- $F_i = False$ signifies that the attack is not possible under the current conditions.

Attack model with rogue control center. In the example of the attack graph involving a rogue control center, we illustrate a Man-in-the-Middle (MITM) attack executed via ARP poisoning. The corresponding logical attack graph is depicted in Figure 5.

The logical relationships within the graph can be expressed using the following Boolean formula:

$$F_1 = (c_1 \vee c_2 \vee p_1) \wedge (c_3 \vee c_4 \vee p_2) \wedge (c_5 \vee c_6). \quad (1)$$

Here, c_i, p_i, and a_i are Boolean variables representing system configurations, attacker privileges, and attack actions, respectively. The operator $\wedge$ denotes logical AND (conjunction), and $\vee$ denotes logical OR (disjunction).

In Figure 5, the graph models an attack on a PLC that combines ARP spoofing with a MITM technique. Specifically:

- c_1: a device(s) with insecure configuration,
- c_2: a device(s) with physical accessibility for the attacker,
- c_3: a communication channel that allows injection of fake messages,
- c_4: a communication path vulnerable to fake instruction injection,
- c_5: a monitoring and detection system that lacks integrity verification of sensor data,
- c_6: a vulnerability that permits triggering critical operating modes due to unauthenticated commands.

The privilege p_1 corresponds to the ability to sniff network traffic and identify the IP addresses of the operator and the PLC. Privilege p_2 represents the ability to impersonate a trusted sender.

The associated attacks include:

- a_1: exploitation of vulnerable devices,
- a_2: use of social engineering or similar methods to gain internal network access,
- a_3: the ARP poisoning attack itself.

The graph (Figure 5) depicts the relationships between system configurations, attacker privileges, and the sequence of actions required to successfully compromise a PLC through ARP spoofing.

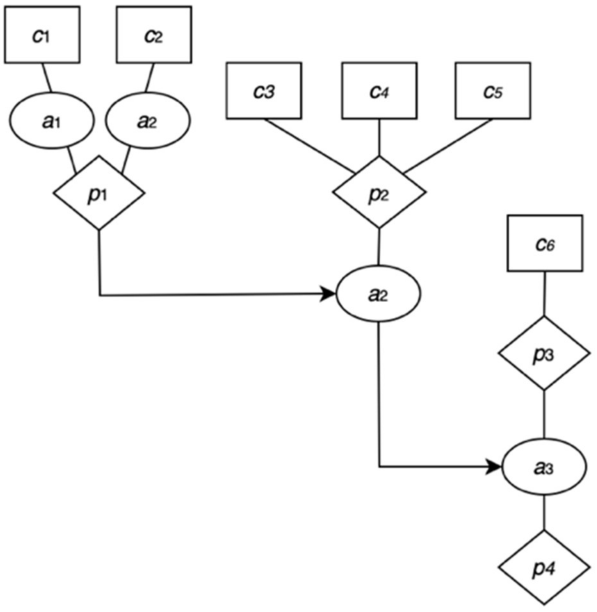


Figure 5: Logical attack graph illustrating an ARP poisoning and Man-in-the-Middle attack scenario.

Rectangles represent system configurations, diamonds denote attacker privileges, and ovals indicate attack actions. Conjunctive and disjunctive logic is used to model dependencies between these elements.

Attack on IFM Safety system. An attack that exploits vulnerabilities in the hardware and software of the existing IFM security system is represented by the logical graph in Figure 6.

In this graph: c_1 denotes a vulnerability to external access; c_2 represents a configuration that is accessible from the local network; c_3 corresponds to insecure administrator access, such as default or weak credentials; c_4 indicates a configuration that permits the execution of system-level scripts, which can dangerously influence control logic; c_5 is a vulnerable configuration allowing external command execution; c_6 refers to a PLC configuration that enables the attacker to trigger overheating or other hazardous states.

The associated privileges are defined as follows:

- p_1: Access to the internal network, such as via default network credentials or insecure services.
- p_2: Access to system configuration files, achieved through administrator-level credentials (often hardcoded or poorly secured).
- p_3: The critical privilege that allows the attacker to inflict physical harm on the system or disrupt its operation.

The attack actions include:

- a_1: Exploitation of device vulnerabilities, allowing the attacker to extract internal system information.
- a_2: Privilege escalation or kernel exploitation of the PLC, leading to deeper logical and physical intrusion into the system.

This attack model is especially relevant for water supply systems and industrial production lines that utilize IFM controllers, where such vulnerabilities can lead to serious safety and operational risks.

Access to configuration files is essential for understanding the internal logic of the system and for making unauthorized modifications to its behavior. The Boolean expression corresponding to the logical attack graph presented in Figure 6 is given by:

$$F_2 = (c_1 \lor c_2 \lor p_1) \land (c_3 \lor c_4 \lor c_5 \lor p_2) \land (p_3 \lor c_6). \quad (2)$$

The graph (Figure 6) models how an attacker can exploit vulnerabilities in network accessibility, administrative access, and PLC configurations to gain critical privileges. Rectangles denote system configurations (e.g., insecure remote access, weak credentials), diamonds represent attacker privileges (e.g., internal network access, control file extraction), and ovals correspond to specific attack actions. The logical structure illustrates how the combination of these factors may lead to system compromise and potential physical harm.

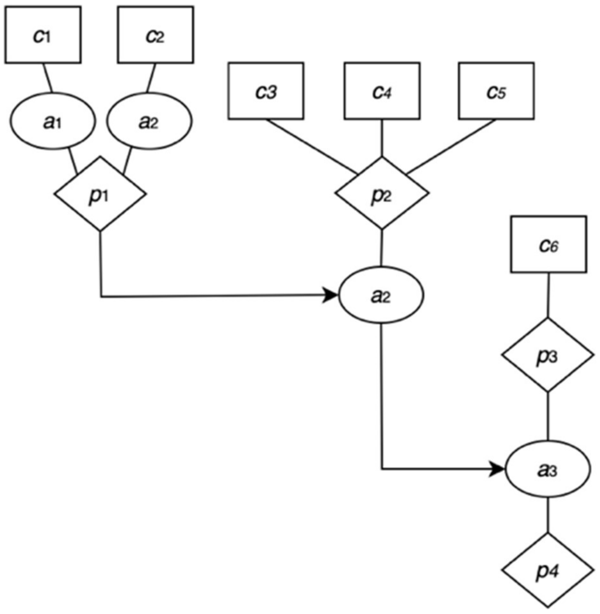


Figure 6: Logical attack graph representing an attack on the IFM Safety system.

The attack model involving CaddyWiper malware. The attack model that leverages CaddyWiper malware can be represented using a logical graph, as illustrated in Figure 7, based on the structured stages of its execution. The corresponding Boolean expression for the attack feasibility is:

$$F_3 = (c_1 \lor c_2 \lor c_3 \lor p_1) \land (c_5 \lor c_4 \lor p_2) \land (c_6 \lor c_7 \lor p_3) \land (c_8 \lor c_9 \lor p_4) \land (c_{10} \lor c_{11} \lor \lor c_{12} \lor p_5). \quad (3)$$

The components used in equation (3) are defined as follows:

- Initial Access and Infection:
  - c_1: Vulnerability to phishing attacks.
  - c_2: Insecure or compromised server account credentials with SCADA access.
  - c_3: Malware infection via SCADA software updates.
  - p_1: Privilege to access SCADA systems.
- SCADA Exploitation:
  - c_4: Critical vulnerabilities in SCADA components.
  - c_5: Vulnerability in the controller's (PLC) web interface.
  - p_2: Privilege to manipulate SCADA data (e.g., hijacking or spoofing).
- Lateral Movement:
  - c_6: Propagation via Windows Management Instrumentation (WMI).
  - c_7: Propagation via Group Policy vulnerabilities.
  - p_3: Privilege to propagate within the internal network.
- Execution and Persistence:

c_8 : Use of PowerShell or VBScript for malware deployment and activation.
c_9: Overwriting of the master boot record (MBR).
p_4: Privilege to execute arbitrary code within SCADA environments.

- Impact and Data Destruction:
  c_{10}: Deletion of configuration files.
  c_{11}: Deletion of system backup copies.
  c_{12}: Deletion of historian data from SCADA systems.
  p_5: Privilege leading to loss of SCADA component control and disruption of industrial processes.

This model reflects a multi-stage, coordinated attack that targets critical infrastructure, demonstrating how combinations of vulnerabilities and acquired privileges can lead to severe operational consequences.

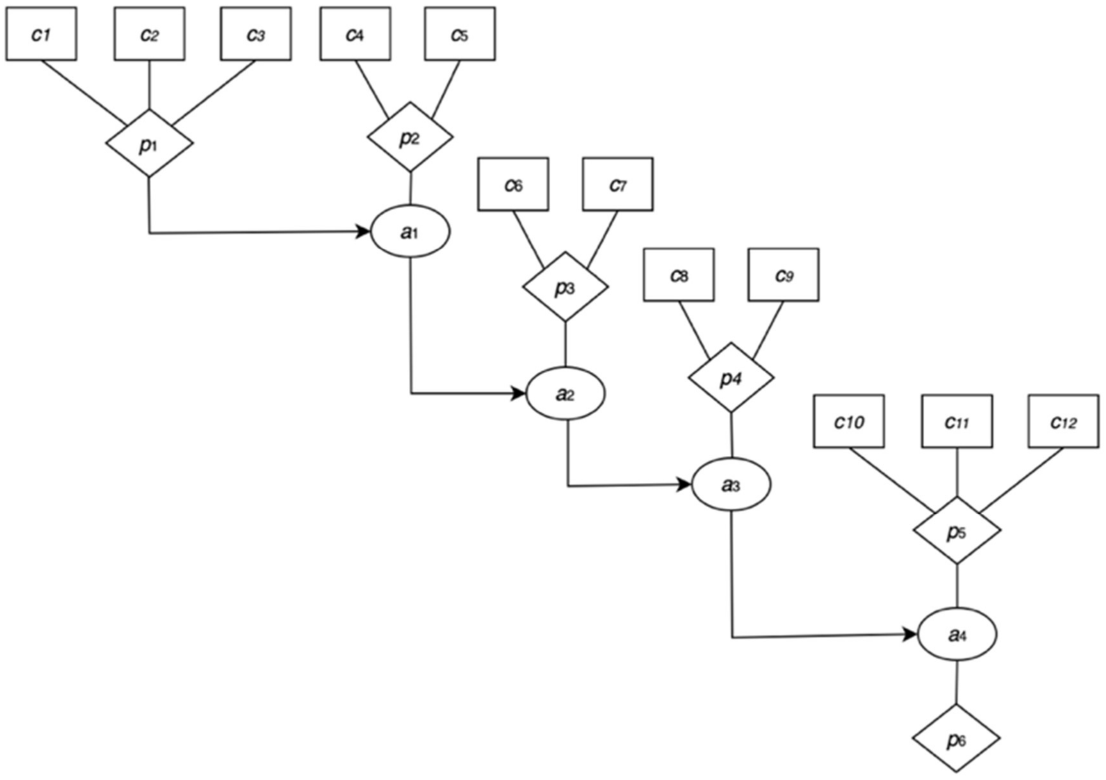


Figure 7: Logical attack graph modeling the execution stages of a CaddyWiper malware-based attack

The graph (Figure 7) illustrates how a combination of vulnerabilities – ranging from phishing and software updates to SCADA misconfigurations – and attacker-acquired privileges can lead to full system compromise. Nodes in the graph represent system configurations (rectangles), attacker privileges (diamonds), and logical relationships among them. The structure captures multiple phases of the attack lifecycle, including initial access, lateral movement, malware execution, and impact on industrial processes, consistent with the Purdue Model.

3.2. Network analysis methodology using graph models and SAT/SMT solvers

We can analyze the formula using SAT/SMT solvers. These solvers require that the expressions be provided in Conjunctive Normal Form (CNF). The expressions F_i must be represented in CNF. For F_1, F_2, F_3 this condition is satisfied.

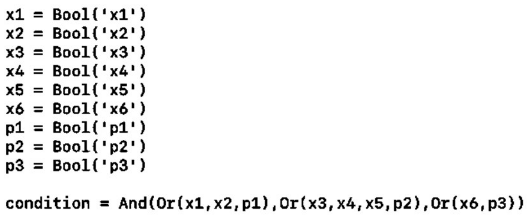
In the computational experiment, the following problems were formulated:

1) Maximize the number of vulnerable configurations that remain active and/or maximize the number of privileges gained by the attacker, such that the expression F_i evaluates to False, under the condition that certain vulnerable configurations always evaluate to True. These correspond

to system vulnerabilities that cannot be mitigated.

2) Identify all combinations of configurations and privileges for which the formula F_i evaluates to True.

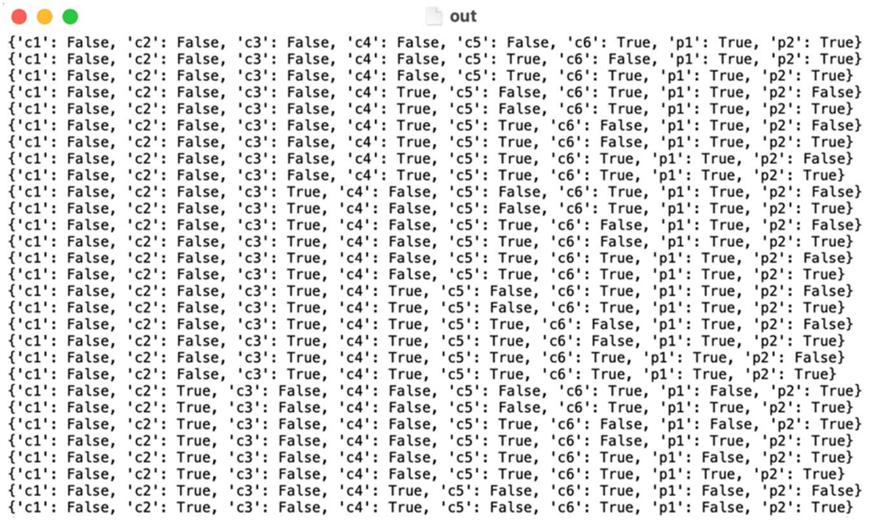
When solving problems 1) and 2), it is necessary to take into account the security policy constraints existing in the system. Such constraints may include: the inability to disable vulnerable services due to usability requirements, the inability to eliminate certain vulnerabilities because doing so would require updated hardware versions, or for other practical limitations. These constraints are also included as part of the SAT/SMT problem formulation. An example of input data is shown in Figure 8.



```
x1 = Bool('x1')
x2 = Bool('x2')
x3 = Bool('x3')
x4 = Bool('x4')
x5 = Bool('x5')
x6 = Bool('x6')
p1 = Bool('p1')
p2 = Bool('p2')
p3 = Bool('p3')

condition = And(Or(x1,x2,p1),Or(x3,x4,x5,p2),Or(x6,p3))
```

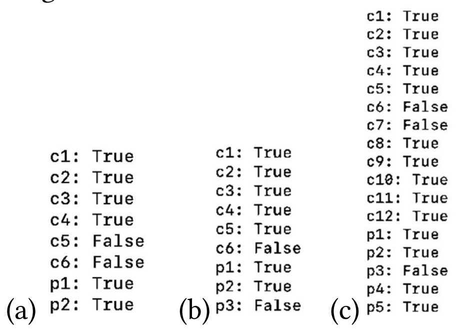
Figure 8: Screenshot of function F_1 input for z3-python.



```
{'c1': False, 'c2': False, 'c3': False, 'c4': False, 'c5': False, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': False, 'c4': False, 'c5': True, 'c6': False, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': False, 'c4': False, 'c5': True, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': False, 'c4': True, 'c5': False, 'c6': True, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': False, 'c4': True, 'c5': False, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': False, 'c4': True, 'c5': True, 'c6': False, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': False, 'c4': True, 'c5': True, 'c6': False, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': False, 'c4': True, 'c5': True, 'c6': True, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': False, 'c4': True, 'c5': True, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': True, 'c4': False, 'c5': False, 'c6': True, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': True, 'c4': False, 'c5': False, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': True, 'c4': False, 'c5': True, 'c6': False, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': True, 'c4': False, 'c5': True, 'c6': False, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': True, 'c4': False, 'c5': True, 'c6': True, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': True, 'c4': False, 'c5': True, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': True, 'c4': True, 'c5': False, 'c6': True, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': True, 'c4': True, 'c5': False, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': True, 'c4': True, 'c5': True, 'c6': False, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': True, 'c4': True, 'c5': True, 'c6': False, 'p1': True, 'p2': True}
{'c1': False, 'c2': False, 'c3': True, 'c4': True, 'c5': True, 'c6': True, 'p1': True, 'p2': False}
{'c1': False, 'c2': False, 'c3': True, 'c4': True, 'c5': True, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': True, 'c3': False, 'c4': False, 'c5': False, 'c6': True, 'p1': False, 'p2': True}
{'c1': False, 'c2': True, 'c3': False, 'c4': False, 'c5': False, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': True, 'c3': False, 'c4': False, 'c5': True, 'c6': False, 'p1': False, 'p2': True}
{'c1': False, 'c2': True, 'c3': False, 'c4': False, 'c5': True, 'c6': False, 'p1': True, 'p2': True}
{'c1': False, 'c2': True, 'c3': False, 'c4': False, 'c5': True, 'c6': True, 'p1': False, 'p2': True}
{'c1': False, 'c2': True, 'c3': False, 'c4': False, 'c5': True, 'c6': True, 'p1': True, 'p2': True}
{'c1': False, 'c2': True, 'c3': False, 'c4': True, 'c5': False, 'c6': True, 'p1': False, 'p2': False}
{'c1': False, 'c2': True, 'c3': False, 'c4': True, 'c5': False, 'c6': True, 'p1': False, 'p2': True}
```

Figure 9: Screenshot of the output file ‘out’ part.

In Figure 9 the excerpt from the SAT/SMT solver output is shown. It is addressing the task of identifying all configuration and privilege combinations that allow the attack to succeed. As shown in Figure 9, the number of configuration combinations that allow the attack to be executed is large when the number of configurations and potential privileges is high. This list can be used to analyze the potential preconditions for a successful attack. Some of the combinations can be filtered out by setting input constraints on configuration values that are not vulnerable (i.e., set to False).



(a)

```
c1: True
c2: True
c3: True
c4: True
c5: False
c6: False
p1: True
p2: True
```

(b)

```
c1: True
c2: True
c3: True
c4: True
c5: True
c6: False
p1: True
p2: True
p3: False
```

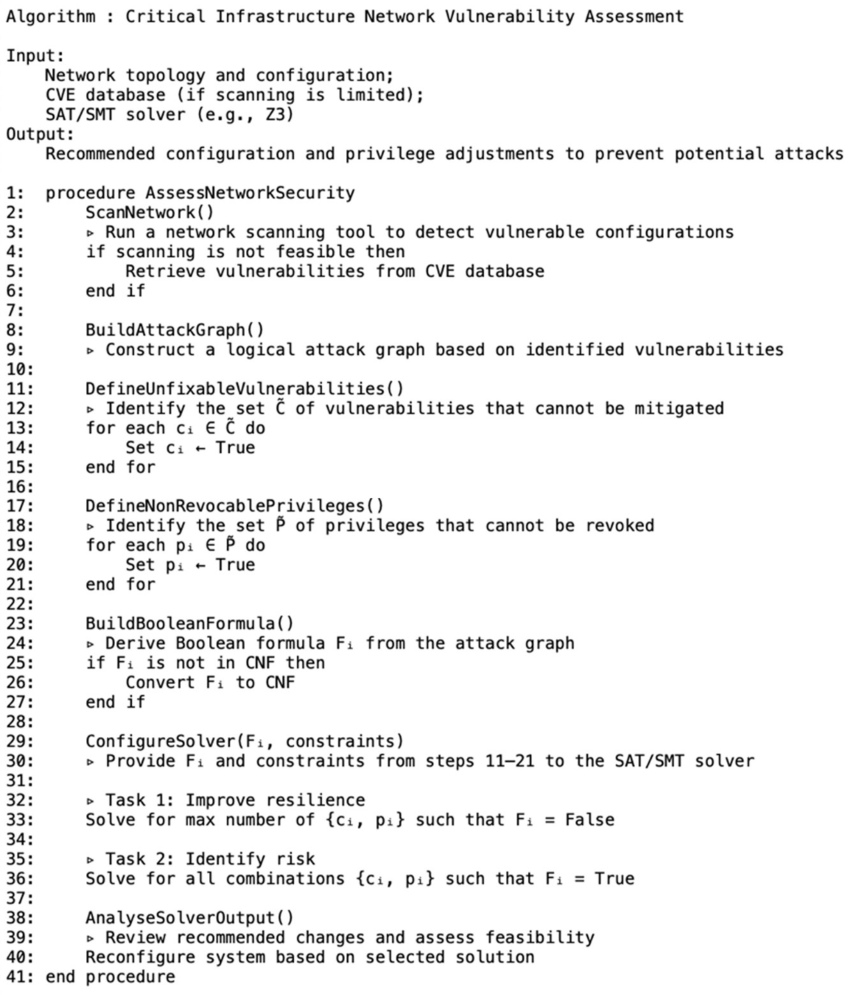
(c)

```
c1: True
c2: True
c3: True
c4: True
c5: True
c6: False
c7: False
c8: True
c9: True
c10: True
c11: True
c12: True
p1: True
p2: True
p3: False
p4: True
p5: True
```

Figure 10: Screenshot of SAT/SMT solver results for identifying the maximum number of vulnerable configurations and privileges under which the corresponding attack fails: (a) attack defined by F_1; (b) F_2; (c) F_3.

The result of identifying the maximum number of configurations and privileges under which the attack becomes infeasible is shown in Figure 10. This type of analysis allows for the identification of specific configurations and privileges that must be adjusted to prevent the attack, without unnecessary reconfiguration of the system.

To prevent the attack based on ARP spoofing (F_1), according to the logic of the SAT/SMT solver, it is sufficient that at least one element in the cyberattack chain – represented by a disjunction within parentheses – fails to hold. This can be minimally achieved by correcting the monitoring and detection subsystem (c_5) – for example, by introducing redundant observation channels or by establishing reference values to detect anomalies artificially created by the attacker. Additionally, it is necessary to prevent the ability to activate critical operating modes that could damage physical components (c_6). This requires the implementation of mechanical safeguards and local control of critical parameters at the device level.



```
Algorithm : Critical Infrastructure Network Vulnerability Assessment

Input:
    Network topology and configuration;
    CVE database (if scanning is limited);
    SAT/SMT solver (e.g., Z3)
Output:
    Recommended configuration and privilege adjustments to prevent potential attacks

1:  procedure AssessNetworkSecurity
2:      ScanNetwork()
3:      ▹ Run a network scanning tool to detect vulnerable configurations
4:      if scanning is not feasible then
5:          Retrieve vulnerabilities from CVE database
6:      end if
7:
8:      BuildAttackGraph()
9:      ▹ Construct a logical attack graph based on identified vulnerabilities
10:
11:     DefineUnfixableVulnerabilities()
12:     ▹ Identify the set $\tilde{C}$ of vulnerabilities that cannot be mitigated
13:     for each $c_i \in \tilde{C}$ do
14:         Set $c_i \leftarrow$ True
15:     end for
16:
17:     DefineNonRevocablePrivileges()
18:     ▹ Identify the set $\tilde{P}$ of privileges that cannot be revoked
19:     for each $p_i \in \tilde{P}$ do
20:         Set $p_i \leftarrow$ True
21:     end for
22:
23:     BuildBooleanFormula()
24:     ▹ Derive Boolean formula $F_i$ from the attack graph
25:     if $F_i$ is not in CNF then
26:         Convert $F_i$ to CNF
27:     end if
28:
29:     ConfigureSolver($F_i$, constraints)
30:     ▹ Provide $F_i$ and constraints from steps 11–21 to the SAT/SMT solver
31:
32:     ▹ Task 1: Improve resilience
33:     Solve for max number of $\{c_i, p_i\}$ such that $F_i$ = False
34:
35:     ▹ Task 2: Identify risk
36:     Solve for all combinations $\{c_i, p_i\}$ such that $F_i$ = True
37:
38:     AnalyseSolverOutput()
39:     ▹ Review recommended changes and assess feasibility
40:     Reconfigure system based on selected solution
41: end procedure
```

Figure 11: Screenshot of algorithm for assessing the security of industrial control systems using logical attack graphs and SAT/SMT solvers.

In other scenarios, a greater number of corrections may be required, which could involve revoking user privileges and potentially conflict with the system's usability policy. In this example, $c_1 = True$, meaning that it is not feasible to eliminate all insecurely configured devices with access to external networks. Similarly, $c_3 = True$ indicates that integrity checks for messages cannot be implemented due to the lack of cryptographic verification tools on relevant devices and in the communication protocols used.

For the attack on the IFM security system (F_2), the minimal corrective strategy involves modifying the vulnerable PLC configuration (c_6) that may expose configuration data and revoking privilege p_3, which allows PLC reconfiguration during runtime. The computation was performed under the constraint $p_1 = True$, which represents access with default credentials. Unfortunately, this cannot be mitigated because the credentials are hardcoded into the devices' firmware.

The general methodology of critical infrastructure network assessment is represented in the form of pseudocode (Figure 11). The Figure 11 outlines the step-by-step process: from modeling attacks and identifying system configurations and privileges, to formulating Boolean expressions, transforming them into Conjunctive Normal Form (CNF), and solving them to evaluate attack feasibility. The output supports decision-making by identifying vulnerable configurations and suggesting minimal corrective actions to enhance system security.

The proposed methodology was validated on three real-world attack models and approved in practical experiments on laboratory testbeds. For each attack, we computed minimal sets of configuration changes required to block the attack using Z3. It is much less than the total number of vulnerable combinations and configurations combinations, which lead to $F_i = true$. These characteristics and solver performance are summarized in Table 2.

Table 2
Summarized results of computer experiment

Attack	Total vulnerable combinations	Minimal fixes required	Solving time (Z3-python) for all vulnerable combinations, s	Solving time (Z3-python) for minimal sets, s
ARP Spoofing and MITM	56	2	0.056749	0.002239
IFM controller vulnerability	180	2	0.139388	0.002247
Caddy Wiper	23520	3	59.087932	0.002832

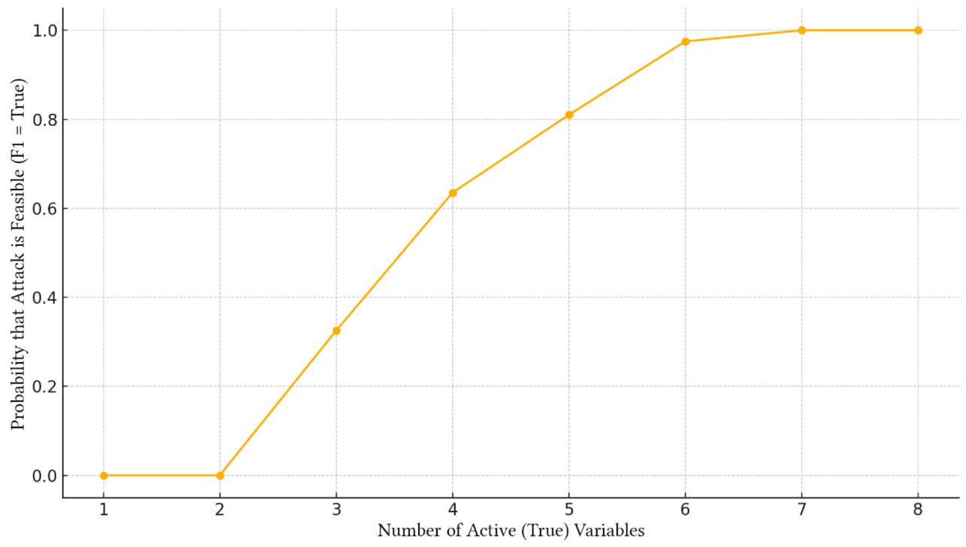
Additionally, using expressions (1)-(3), we can analyze dependence of attack feasibility probability on the number of vulnerable configurations and privileges $\{c_i, p_i\}$, which characterizes the general system resilience. The results are presented in Figure 12 for F_1 and F_3 respectively. Each point represents the fraction of true Z3 outcomes over 100 random samples with k active variables.

The scripts developed for conducting the computer experiment are presented in [38].

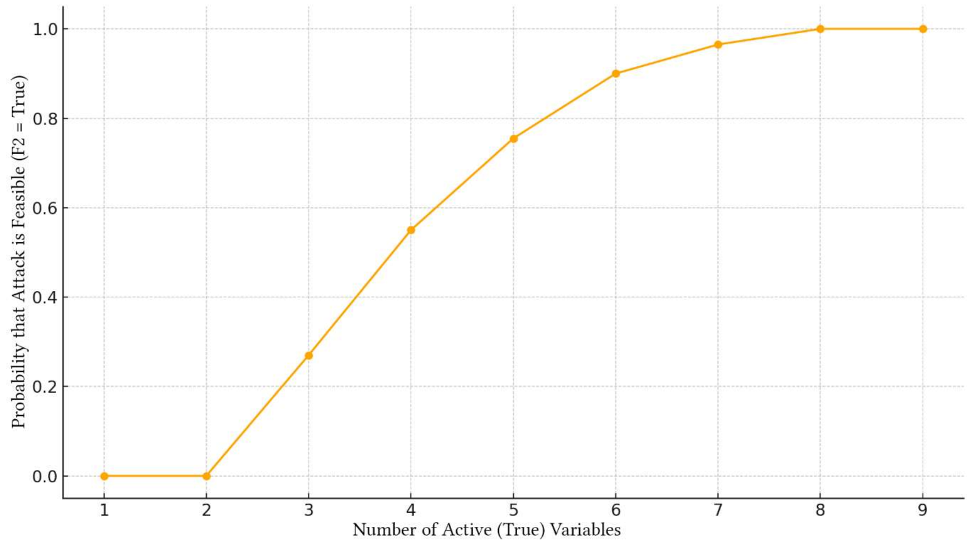
By reviewing the solver's suggestions, a system administrators can form their own judgment about which changes should be implemented. The solver provides a formal estimate of the minimal set of configurations and/or privileges that need to be modified to render the attack infeasible. However, due to financial or other practical considerations, it may be more feasible to apply alternative changes. Therefore, the administrator should also analyze other viable options. The SAT/SMT solver facilitates this process by producing a structured list of all vulnerable configuration combinations that make the attack possible, enabling the selection of the most appropriate variant for system reconfiguration.

4. Conclusions

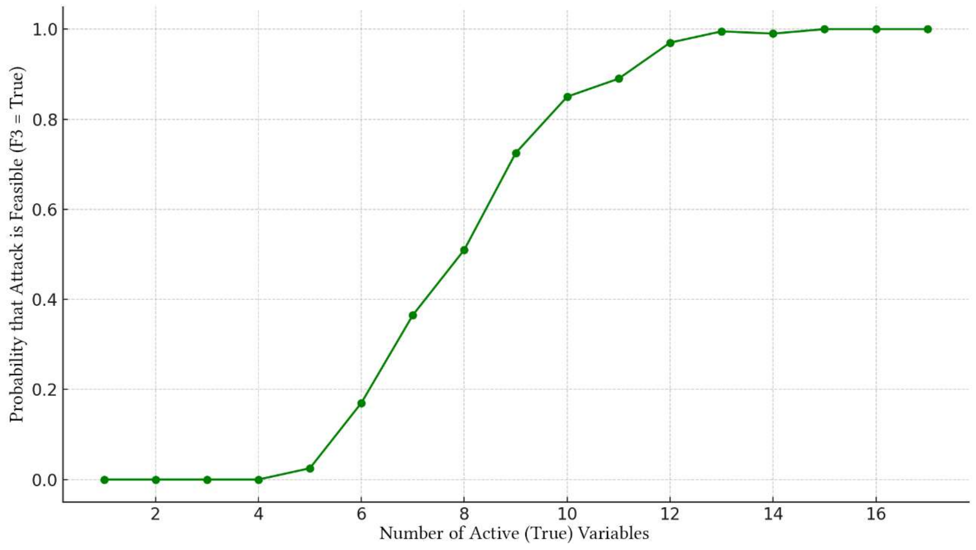
The proposed models, constructed in the form of logical graphs, enable a comprehensive reconstruction of cyberattacks, allowing for the analysis of their impact on ICS components. These models facilitate the identification of weak configurations and the necessary conditions that enable successful attacks. An analysis of typical attack graphs across various ICS elements reveals that reducing the attack surface requires minimizing the number of communication channels that could potentially be exploited. Additionally, the number of vulnerable devices should be reduced by upgrading them to more secure alternatives. Wherever feasible, cryptographic mechanisms should



(a)



(b)



(c)

Figure 12: Probability of successful attack realization (F_i = True) depending on the number of active (vulnerable) system configurations and privileges: (a) F_1, (b) F_2, (c) F_3.

be employed to ensure authentication and integrity of control and monitoring messages. Physical safeguards, such as mechanical fuses, should also be considered to prevent systems from entering unsafe operational states.

The proposed methodology allows for the detection of critical system configurations and provides a list of minimally required changes to block potential attack paths. The integration of SAT/SMT solvers into this process enables automation of the analysis and configuration evaluation. The developed solver scripts serve as a practical tool to assist security administrators in making informed decisions.

Ultimately, timely reconfiguration of ICS environments, based on the identified attack vectors, enables proactive defense. This approach helps to anticipate possible infection, propagation, and persistence mechanisms used by advanced malware, thereby strengthening the system's overall cybersecurity posture.

Acknowledgements

We thank the students of the Institute of Physics and Technology – Yurii Prykhodko, Oleksii Bondarenko, and Dmytro Krygin – for their assistance with the laboratory testbed experiments.

Declaration on Generative AI

During the preparation of this work, the authors used GPT-4o in order to make grammar and style check. After using this tool, the authors reviewed and edited the content and take full responsibility for the publication's content.

References

[1] T. Alladi and V. Chamola, Industrial Control Systems: Cyberattack Trends and Countermeasures, Computer Communications, 2020. doi: 10.1016/j.comcom.2020.03.007.
[2] A. M. Mohan, N. Meskin, and H. Mehrjerdi, A Comprehensive Review of the Cyber-Attacks and Cyber-Security on Load Frequency Control of Power Systems, Energies, vol. 13, no. 15, p. 3860, 2020. doi: 10.3390/en13153860.
[3] A. Dumitracu et al., Environment Communication and Control Systems Integrated on Teaching Platforms. Case Study: Double Water Tank System, in Proc. 20th Int. Conf. on Control Systems and Science, 2015, pp. 946–951.
[4] O. Novikov, M. Shreider, I. Stopochkina, and M. Ilin, Cyber Attacks Simulation for Modern Energy Facilities, in CEUR Workshop Proceedings, Selected Papers of the XXIII International Scientific and Practical Conference "Information Technologies and Security" (ITS 2023), pp. 35–49. [Online]. Available: https://ceur-ws.org/Vol-3887/paper4.pdf.
[5] G. Endelberg, Process-Aware Attack-Graphs for Risk Quantification and Mitigation in Industrial Infrastructures, in CEUR Workshop Proceedings, vol. 3139, 2022. [Online]. Available: https://ceur-ws.org/Vol-3139/paper02.pdf.
[6] O. Sheiner and J. Wing, Tools for Generating and Analyzing Attack Graphs, in Formal Methods for Components and Objects, de Boer et al., Eds., LNCS 3188, Springer, 2004, pp. 344–371.
[7] A Logic-Based Enterprise Network Security Analyzer. [Online]. Available: https://github.com/risksense/mulval.
[8] X. Ou, S. Govindavajhala, and A. W. Appel, MulVAL: A Logic-Based Network Security Analyzer, in Proc. 14th USENIX Security Symp., 2005. [Online]. Available: https://www.usenix.org/legacy/event/sec05/tech/full_papers/ou/ou.pdf.
[9] D. Catta, J. Leneutre, A. Mijatovic, J. Ulin, and V. Malvone, A Formal Verification Approach to Handle Attack Graphs, in Proc. 16th Int. Conf. on Agents and Artificial Intelligence (ICAART 2024) – Vol. 3, pp. 125–132. doi: 10.5220/0012310000003636. [Online]. Available: https://www.scitepress.org/Papers/2024/123100/123100.pdf.

[10] F. Ö. Sönmez, C. Hankin, and P. Malacaria, Attack Dynamics: An Automatic Attack Graph Generation Framework Based on System Topology, CAPEC, CWE, and CVE Databases, Computers & Security, vol. 123, p. 102938, Dec. 2022. doi: 10.1016/j.cose.2022.102938.
[11] CoReS: A Tool for Computing Core Graphs via SAT/SMT Solvers, Journal of Logical and Algebraic Methods in Programming, vol. 109, no. 3/4, p. 100484, Aug. 2019. doi: 10.1016/j.jlamp.2019.100484.
[12] J. Homer and X. Ou, SAT-Solving Approaches to Context-Aware Enterprise Network Security Management, IEEE J. Sel. Areas Commun., vol. 27, no. 3, Apr. 2009. doi: 10.1109/JSAC.2009.090407.
[13] S. Jha, O. Sheyner, and J. Wing, Two formal analyses of attack graphs, in Proc. 15th IEEE Computer Security Foundations Workshop (CSFW-15), Cape Breton, NS, Canada, 2002, pp. 49–63. doi: 10.1109/CSFW.2002.1021806.
[14] L. Davis and T. Ji, Evaluating SAT and SMT Solvers on Large-Scale Sudoku Puzzles, 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2501.08569.
[15] J. K. Fichte, D. Le Berre, M. Hecher, and S. Szeider, The Silent Revolution of SAT, Communications of the ACM, 2023. [Online]. Available: https://cacm.acm.org/research/the-silent-revolution-of-sat/.
[16] Z3-solve. [Online]. Available: https://pypi.org/project/z3-solver/.
[17] Kyiv Polytechnic University, USAID, and Partners Strengthen Ukraine's Cyber Resilience. [Online]. Available: http://icslab.kpi.ua/.
[18] S7 Communication between SIMATIC S7-1200 and SIMATIC S7-300. [Online]. Available: https://cache.industry.siemens.com/dl/files/951/92269951/att_54641/v5/s7communication_s7-1200_s7-300_en.pdf.
[19] Python-snap 7. [Online]. Available: https://pypi.org/project/python-snap7/.
[20] Telnet protocol specification. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc854].
[21] IFM: Vulnerabilities in ifm AC14 Firmware. [Online]. Available: https://certvde.com/de/advisories/VDE-2024-012/.
[22] IFM SMART PLC AC14XX/SMART PLC AC4XXS up to 4.3.17 Hard-Coded Credentials. [Online]. Available: https://vuldb.com/?id.270585.
[23] Raw Mode PTY Echo Race Condition Privilege Escalation. [Online]. Available: https://www.exploit-db.com/exploits/33516.
[24] CaddyWiper. [Online]. Available: https://attack.mitre.org/software/S0693/.
[25] D. Iuzvyk and T. Peck, SECURONIX Threat Labs Initial Coverage Advisory: INDUSTROYER2/CADDYWIPER Targeting Ukrainian Power Grid – Detailed Analysis. [Online]. Available: https://www.securonix.com/blog/industroyer2-caddywiper-targeting-ukrainian-power-grid/.
[26] ICS Matrix. [Online]. Available: https://attack.mitre.org/matrices/ics/.
[27] Spearphishing Attachment. [Online]. Available: https://attack.mitre.org/techniques/T0865/.
[28] External Remote Services. [Online]. Available: https://attack.mitre.org/techniques/T0822/.
[29] Supply Chain Compromise. [Online]. Available: https://attack.mitre.org/techniques/T0862/.
[30] Exploit Public-Facing Application. [Online]. Available: https://attack.mitre.org/techniques/T0819/.
[31] Valid Accounts. [Online]. Available: https://attack.mitre.org/techniques/T0859/.
[32] Replication Through Removable Media. [Online]. Available: https://attack.mitre.org/techniques/T0847/.
[33] Scripting. [Online]. Available: https://attack.mitre.org/techniques/T0853/.
[34] System Firmware. [Online]. Available: https://attack.mitre.org/techniques/T0857/.
[35] Data Destruction. [Online]. Available: https://attack.mitre.org/techniques/T0809/.
[36] Denial of Control. [Online]. Available: https://attack.mitre.org/techniques/T0813/.
[37] Indicator Removal on Host. [Online]. Available: https://attack.mitre.org/techniques/T0872/.
[38] ICS Attack Graph Models – GitHub Repository, 2025. [Online]. Available: https://github.com/infosec-kpi/ics-attack-lab-models.