Towards Integration of EPANET and ASM2S To Enhance Security in Water Distribution Systems

George E. Raptis
Industrial Systems Institute (ISI)
ATHENA RC
Patras, Greece
graptis@isi.gr

Muhammad Taimoor Khan
School of Computing and Mathematical Sciences
University of Greenwich
London, United Kingdom
m.khan@greenwich.ac.uk

Christos Koulamas
Industrial Systems Institute (ISI)
ATHENA RC
Patras, Greece
koulamas@isi.gr

Dimitrios Serpanos
Electrical and Computer Engineering, University of Patras,
Computer Technology Institute and Press "Diophantus", &
Industrial Systems Institute (ISI)
Patras, Greece
serpanos@ece.upatras.gr

***Abstract*—In the decentralized Industrial Control Systems (ICS) era, water distribution systems (WDS) are critical in ensuring water safe and reliable delivery. However, their growing complexity, connectivity, and distributed nature expose them to cybersecurity risks. Renowned WDS software, like EPANET, lacks features to address such risks, which, however, can be addressed by complementary solutions, like ASM2S. In this paper, we compare the capabilities offered by these two tools and make a first step towards exploring their combination, aiming to equip WDS tools with enhanced hydraulic, water quality, and cybersecurity modeling and monitoring characteristics.**

***Index Terms*—distributed industrial control systems (ICS), cybersecurity, critical systems, behavior-based security monitoring, water management system, computational attacks, network attacks, false data injection (FDI) attacks**

I. Introduction

Industrial Control Systems (ICS) manage and supervise critical infrastructure in the industrial automation and control landscape. These systems, which integrate hardware and software with network technologies, control numerous processes across multiple sectors, including water management, power generation, etc. Traditionally centralized, the architecture of ICS has been progressively shifting towards a more distributed model to enhance resilience, scalability, and real-time response capabilities.

Water distribution systems (WDS) are a crucial domain of distributed ICS, ensuring safe and reliable water delivery. WDS use a variety of sensors and actuators managed by control software to monitor and handle water distribution events (e.g., flow rates and pressure changes throughout the distributed network). However, as WDS become more interconnected, complex, and decentralized, they also become more susceptible to cybersecurity threats that can compromise the safety and operation of the water distribution systems.

To address these vulnerabilities, in their recent research, Raptis et al. [1] introduced the ASM2S inline security monitoring approach to monitor system behavior and detect anomalies in real time, enhancing the security of WDS by identifying and mitigating potential cyber threats. Security monitoring depends on the expressiveness of the models used; such capabilities are not provided in EPANET [1], a well-established software application for simulating water distribution networks, which provides extensive features for hydraulic and water quality modeling. Failing to monitor security and recover the system (i.e., resilience) from attacks means that such tools cannot ensure the continuous operation of critical infrastructures, which is a key requirement.

Recognizing the strengths and limitations of both systems (EPANET and ASM2S), this paper compares the capabilities of these systems and, given their complementary nature, it makes a first step towards exploring the integration of ASM2S's security features into EPANET's modeling framework, aiming to provide an integrated tool that not only supports advanced hydraulic and quality simulations but also ensures the secure and uninterrupted operation of water distribution systems in the face of evolving cybersecurity challenges.

A. Objective

The paper's objective is two-fold: i) compare and contrast the capabilities of ASM2S and EPANET in managing WDS features (hydraulic modeling, water quality modeling, security and resilience modeling) and ii) explore the integration of ASM2S's cybersecurity features into the EPANET framework. By meeting this objective, we make a first step towards combining EPANET's extensive hydraulic and water quality modeling features with ASM2S's real-time security monitoring, addressing the increasing need for cybersecurity in distributed ICS. The overall objective is to seek ways to create complete and secure water distribution management

[1]Official EPANET website: https://www.epa.gov/water-research/epanet; Github repository: https://github.com/USEPA/EPANET2.2

solutions, enhancing the functionality and safety of these critical infrastructures.

B. Paper structure

The remainder of the paper is organized as follows. In Section II, we present related works focusing on security monitoring in distributed ICS and applying security approaches in EPANET. In Section III, we compare EPANET and ASM2S WDS, focusing on their capabilities in hydraulic modeling, water quality modeling, and security and resilience modeling. In Section IV, we make the first steps towards exploring the integration of ASM2S security features in EPANET. We present the threat model, focus on specific security properties (deadlocks and livelocks), introduce the inline security monitoring approach, and provide examples. In Section V, we discuss the main takeaways of the work presented and provide future research directions.

II. Related Works

The use of distributed ICS has increased in recent years, highlighting the significance of security monitoring. As the complexity and connectivity of these systems increase, they become more vulnerable to cyber attacks, making real-time monitoring of their security properties essential for identifying and mitigating potential threats. Typically, run-time security monitors define a reference behavior and activate alarms when the observed behavior matches known bad behavior or deviates from expected good behaviors. Hence, specifying the reference behavior is a critical characteristic of a security monitor.

Security monitoring and analytics tools in distributed ICS are important, although deployment and utilization can be complex [2]. Several research attempts have been made in this direction [3], proposing that advanced technologies can be adopted in distributed ICS, such as using Artificial Intelligence (AI) to detect and respond to security threats in real time [4], applying aggregation and normalization methods to handle heterogeneous data [5], adding features to existing architectures [6], providing data desensitization and enhanced threat visualization [7], and adopting variable-specific prediction models [8].

Focusing on distributed WDS, Raptis et al. [1] emphasized the importance of run-time security monitoring to address challenges like communication delays and potential cyber threats, by proposing an approach (ASM2S) that specifies security properties within distributed environments, enhancing the system's reliability and security through real-time monitoring. Their approach implements inline security monitoring techniques, extending other works [9, 10] in the domain. The proposed approach directly generates run-time checks of WDS behavior, with any violation indicating a potential security incident, thus, activating the appropriate alarm.

Focusing on WDS and EPANET software, researchers have used EPANET to simulate various scenarios of cyber-physical attacks, thereby identifying vulnerabilities and potential impacts on water distribution systems. For example, Taormina et al. [11] used EPANET to simulate the effects of malicious cyber-physical attacks, providing valuable insights into how such breaches could compromise the safety and functionality of water distribution systems; similarly, Karrenberg et al. [12] explored using EPANET alongside finite state processes to pinpoint vulnerabilities, demonstrating the utility of hydraulic models in cybersecurity assessments.

Other works focus on extending EPANET capabilities. For example, Taormina et al. [13] developed epanetCPA, a toolkit that extends EPANET's capabilities by assessing water networks' hydraulic responses to cyber attacks, enabling high-fidelity simulations and enhancing water utilities' predictive accuracy and preparedness against potential cyber threats. Similarly, Mahmoud et al. [14] introduced WDSchain, which processes time-series data from EPANET supporting simulations with static and dynamic blockchain modes to improve security and automation within WDS.

EPANET has also been used for cybersecurity research and education by leveraging the generation and analysis of ICS datasets [15, 16] and the use of the platform to build cyber modeling and testing toolkits [17, 18]. We should also note that the vulnerability of WDS to cyber-physical threats was exemplified by the Stuxnet incident, where highly sophisticated malware specifically targeted ICS. Such implications underscore the potential risks facing similar infrastructures that depend on software like EPANET for operational integrity [19].

The related works underscore the importance of security monitoring in distributed ICS, particularly within the WDS domain through the ASM2S approach, and highlight that well-established simulation WDS tools, like EPANET, have been used for cybersecurity ICS research. However, it is evident that there are limitations regarding EPANET security monitoring capabilities; thus, in the next section, we first analyze and compare the capabilities offered by these two tools (EPANET and ASM2S WDS), and then we explore the first steps towards combining them.

III. EPANET versus ASM2S WDS

EPANET (Fig. 1) is a software application developed by the Environmental Protection Agency (EPA) to simulate water movement and quality behavior within pressurized pipe networks. Engineers and planners widely use this tool to design and optimize water distribution systems. EPANET performs extended-period simulations of the hydraulic and water quality behavior within water distribution piping systems, tracking the flow of water in each pipe, the pressure at each node, the height of water in each tank, and the concentration of chemical species throughout the network. In their recent research, Raptis et al. [1] introduced the ASM2S WDS (Fig. 2) that implements run-time securing monitoring within distributed ICS environments specifying security properties (e.g., deadlocks, livelocks) and adopting a core WDS structure. In the following subsections, we compare these tools regarding hydraulic modeling, water quality modeling, and security and resilience modeling capabilities. Table I summarizes the main capabilities of these tools.

A. Hydraulic modeling

Regarding hydraulic modeling, EPANET's capability to use pressure-dependent demands ensures that hydraulic analyses reflect real-world conditions, enabling system fine-tuning for efficiency and reliability. EPANET supports control options, ranging from simple tank level and timer controls to intricate rule-based controls, which are essential for managing system operations effectively. Moreover, the software's ability to analyze networks of unlimited size is beneficial for large-scale projects, removing constraints on system expansion and complexity. Further enhancing its utility, EPANET computes friction head losses using well-known formulas such as Hazen-Williams, Darcy-Weisbach, and Chezy-Manning, and it includes minor head losses for various fittings and bends. This modeling is crucial for optimizing system design and reducing unnecessary energy consumption. Hydraulic modeling supports constant and variable speed pumps, providing flexibility in system design and operation, and calculates the associated energy costs, aiding in financial planning and budgeting. Moreover, EPANET's versatility extends to modeling different types of valves (e.g., shutoff, check, pressure regulating, and flow control), which are necessary for maintaining system integrity and functionality. Its capability to allow storage tanks of any shape and to consider multiple demand categories at nodes with distinct time variation patterns offers unparalleled customization and precision in system modeling. Lastly, modeling pressure-dependent flow from emitters further refines the simulation of real-world conditions, ensuring that the system's design can handle variable demand scenarios effectively.

ASM2S WDS [1] offers a limited body of capabilities compared to EPANET. In particular, it supports core functionalities regarding hydraulic modeling by adjusting system operation based on simple tank levels or timer control functions, enabling analysis of WDS networks of unlimited size, providing sensors and actuators (e.g., pumps and valves) with constant or dynamic speed behavior, and, considering different water demand levels and commands (e.g., fill or drain) at different nodes and tanks. Moreover, it partially supports the computation of pumping energy and cost and the modeling of diverse components' characteristics.

B. Water quality modeling

Regarding water quality modeling, EPANET offers several capabilities ensuring the safety and integrity of WDS. It models storage tanks using different reactor types (e.g., complete mix, plug flow, and two-compartment), allowing for accurate simulation of how water quality changes within tanks under diverse conditions. Moreover, it can track the movement of non-reactive and reactive materials through the WDS network, capturing how substances remain stable or change (grow or decay) over time, aiming to improve the prediction of contaminant spread and implementing safety measures. Additionally, EPANET calculates the water age throughout a WDS, which is required to manage water freshness and prevent stagnation. Determining the percentage of flow from a specific node reaching all other nodes over time

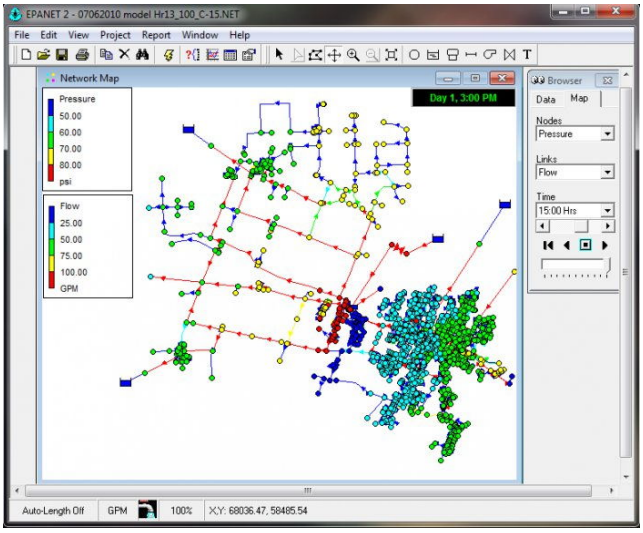

Fig. 1. Distributed water network used in EPANET

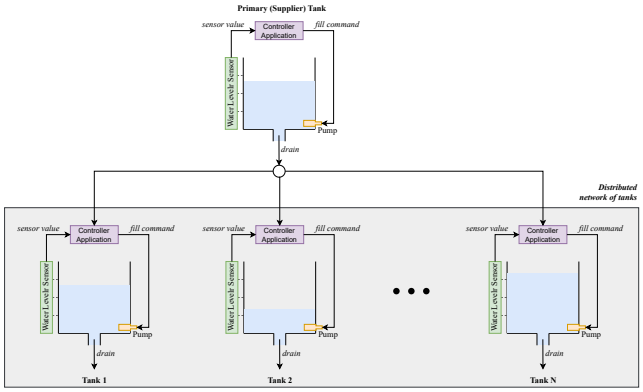

Fig. 2. Distributed water management system used in ASM2S WDS

enhances understanding of water distribution dynamics and system vulnerabilities. EPANET also considers both bulk flow and pipe wall reactions, including the effects of mass transfer limitations; these are essential for the realistic modeling of chemical interactions within the WDS network. Furthermore, EPANET enables the simulation of growth or decay reactions up to a limiting concentration, and it correlates wall reaction rate coefficients with pipe roughness, providing a nuanced approach to understanding how infrastructure condition affects water quality. The flexibility to input varying concentrations or mass inputs at any network location allows for dynamic simulation of different scenarios, enhancing the tool's utility for planning and emergency response.

Similarly to hydraulic modeling, ASM2S WDS [1] offers limited capabilities compared to EPANET regarding water quality modeling. In partcular, ASM2S WDS can calculate the age water throughout the simulation, distributes portions of water from a given set of nodes to a target collection of other nodes, and allows for different commands and functions as inputs at different location in the WDS network. Moreover,

TABLE I
MODELING AND MONITORING CAPABILITIES OF EPANET AND ASM2S WDS TOOLS

Modeling and Monitoring Capability	EPANET	ASM2S WDS
Hydraulic		
Use pressure-dependent demands in hydraulic analyses	●	○
System operation based on simple tank level or timer controls and complex rule-based control	●	●
No limit on the size of the network that can be analyzed	●	●
Computes friction headloss using the Hazen-Williams, Darcy-Weisbach, or Chezy-Manning formulas	●	○
Includes minor head losses from bends, fittings, etc.	●	○
Models constant or variable speed pumps	●	●
Computes pumping energy and cost	●	◐
Models various types of valves, including shutoff, check, pressure regulating, and flow control	●	◐
Allows storage tanks to have any shape	●	○
Considers multiple demand categories at nodes, each with its own pattern of time variation	●	●
Models pressure-dependent flow issuing from emitters	●	○
Water Quality		
Storage tanks as being either complete mix, plug flow, or two-compartment reactors	●	◐
Movement of a non-reactive tracer material through the network over time	●	○
Movement and fate of a reactive material as it grows or decays with time	●	○
Age of water throughout a network	●	●
Percent of flow from a given node reaching all other nodes over time	●	●
Reactions in the bulk flow and at the pipe wall	●	○
Accounts for mass transfer limitations when modeling pipe wall reactions	●	◐
Allows growth or decay reactions to proceed up to a limiting concentration	●	○
Allows wall reaction rate coefficients to be correlated to pipe roughness	●	○
Allows for time-varying concentration or mass inputs at any location in the network	●	●
Security and Resilience		
System and network invariant	○	●
Supports real-time monitoring of water distribution networks	○	●
Modeling and monitoring of known threats	○	●
Defines security properties to enhance real-time monitoring	○	●
Addresses critical challenges found in distributed systems such as deadlock and livelock	○	●
Detects various types of security threats	○	●
Identifies potential vulnerabilities and assesses their likelihood and consequences	○	●
Provides decision support to optimize operation, improve resource utilization, and enhance resilience	○	●

● fully supported; ◐ partially supported; ○ not supported

it partially offers capabilities like diversifying reactor types for storage tanks and considering mass transfer limitations during the modeling process. As we see, ASM2S WDS lacks several capabilities offered by EPANET.

C. Security and resilience modeling

While EPANET excels in hydraulic and water quality modeling, it has security and resilience modeling limitations. EPANET supports such features indirectly; for example, by modeling different scenarios (e.g., pipe breaks and pump failures), EPANET allows users to analyze the resilience of WDS under various conditions. In this direction, extensions to EPANET can simulate the interactions between multiple agents and assets (e.g., chemical and biological agents, bulk water, and pipe walls) in WDS. However, EPANET and its extensions (e.g., EPANET-MSX and EPANET-RTX) are not designed for security per se (e.g., monitoring threats or automatically responding to security incidents). However, its simulation capabilities are essential for planning and improving the security and resilience of WDS.

On the other hand, ASMS2S WDS [1] offers a wide range of security and resilience modeling capabilities. First, it supports real-time monitoring of water distribution networks, which is crucial for quick detection and response to security incidents. Moreover, ASM2S WDS enables the specification of security properties to enhance the effectiveness of real-time monitoring. These security properties define conditions arising from the distributed nature of WDS and could give rise to cybersecurity incidents. ASM2S WDS monitor can address critical challenges inherent in distributed systems, such as deadlock and livelock, ensuring smooth and continuous WDS operation. Deadlocks in WDS can result in operational standstills, preventing effective water management and distribution and, thus, jeopardizing system reliability and efficiency. Similarly, livelocks in WDS can cause continuous but non-progressive activity, wasting resources and energy while failing to achieve WDS progress or stability. Towards this direction, ASM2S

WDS can detect various security threats, identify potential vulnerabilities, and assess their likelihood and consequences. Through these capabilities, ASMS2S WDS provides decision support, helping operators optimize operations, improve resource utilization, and enhance the resilience of WDS against a range of threats. Hence, ASM2S WDS aims to enhance the security and resilience of WDS, ensuring they are better equipped to manage the complexities of modern threats and operational challenges.

IV. First Steps towards Integration

As we can see, EPANET offers a wide range of capabilities in hydraulic and water quality modeling, while ASM2S specializes in security and resilience modeling. Integrating these capabilities would provide a comprehensive and robust solution for WDS. In this section, we take the first steps toward this direction.

A. Threat model

In a distributed WDS, programs (e.g., controllers) are composed of instructions operating and communicating on external input data values. These components are vulnerable to three main threat types: i) computational attacks, ii) network attacks, and iii) false data injection (FDI) attacks. This paper focuses on computational attacks, as they target the core WDS functionality by exploiting vulnerabilities and potentially leading to a complete system compromise. In a computational attack, an adversary attempts to modify the program's functional or non-functional behavior. In the first case, an adversary can i) modify the instructions or internal data values of the implementation or execution across WDS, ii) exploit vulnerabilities or bugs across multiple interconnected WDS components leading to non-desired situations (e.g., buffer overflows and memory corruption), and iii) inject additional code into program implementation or execution across WDS components. Regarding the non-functional behavior, the adversary can affect it by running another program(s) on one or more WDS components, which may affect the performance of the entire WDS. ASM2S specifies the WDS program's functional and non-functional behavior to detect potential threats. By comparing the actual behavior with its specified behavior, ASM2S can identify such potential computational attacks.

B. Security properties

We will focus on the security properties of deadlocks and livelocks, raised by the decentralized nature of WDS [1]. In WDS, a deadlock can occur when multiple control subsystems managing different network parts simultaneously rely on each other's operations to proceed. For example, if one subsystem controls water flow based on the pressure readings from another subsystem, and this second subsystem, in turn, adjusts its pressure control based on the flow rate outputs of the first, a circular dependency could form. If each subsystem waits for the other to complete its action before proceeding with its adjustments, neither can advance, leading to a deadlock where water flow and pressure management come to a standstill. A livelock can occur when multiple control subsystems continuously adjust their operations in response to each other's changes without reaching an acceptable operational state. For example, one subsystem adjusts the flow rate in a pipeline based on real-time water demand data, while another subsystem simultaneously modulates the pressure based on the flow rates. If these adjustments become cyclically dependent, each subsystem repeatedly alters its settings in response to changes made by the other without stabilizing, and the WDS enters a state of constant activity with no progress. This continuous but unproductive loop prevents WDS from reaching operational stability, wasting resources and reducing efficiency.

C. Run-time security monitoring

We follow the ASM2S approach [1] to apply model-based inline run-time security monitoring in WDS. This approach allows for fast and efficient checking of program behavior at run-time, with any violation of an assertion (i.e., construct that verifies that a specified condition holds true at a particular point in the code) indicating a potential security incident. This is a four-step approach: i) the WDS program, which is annotated with a formal specification language (e.g., JML for Java-based applications) to describe the behavior of the program, is compiled using a modified compiler that produces both the executable code and a separate file containing the translated assertions, ii) the executable code and the corresponding assertion file are deployed on the WDS components, iii) during the operation of the WDS application (i.e., run-time), the inline monitor executes the assertions alongside the executable code, and iv) if an assertion fails, the inline monitor raises an alarm indicating a security violation. The following sub-section explores integrating ASM2S security features with EPANET source code.

D. Exploring integration

To explore the capability of integrating ASM2S and EPANET, we used the Java-based EPANET engine implementation [2]. At first, we present a basic example (Listing 1) of how a run-time security monitor can use a formal specification language (JML for Java applications) along with proposed annotations. The `normal_behavior` annotation specifies the basic sanitary checks, while the `compromised_behavior` is an extended annotation that determines the ICS behavior against the defined cyber-attack classes. The `requires` clause of `compromised_behavior` defines the condition represented through `jml-spec-expr`. The main difference between `requires` and `alarms` clauses is the generated code, as the run-time security monitor only generates assertions against the `requires` clause. In contrast, the `alarms` clause generates code to document the ongoing attack and the available metadata (e.g., failed condition, module name, time of the attack).

[2]Github repository for Java-based EPANET: https://github.com/Baseform/Baseform-Epanet-Java-Library/

Listing 1. Example of using JML to specify security properties

```
normal-beh ::= normal_behavior
    requires jml-spec-expr;
    ensures jml-spec-expr;

compromised-beh ::= also compomised_behavior
    requires jml-spec-expr;
    ensures jml-spec-expr;
    alarms (Threat) jml-spec-expr;
```

Listing 2 provides an example of specifying normal and compromised behavior for `createSimulationNetwork` method, which creates a WDS network for hydraulic modeling. The `requires` clause specifies the preconditions that must be met before the method is executed, defining the state that the system or the variables must be in for the `createSimulationNetwork` method to run correctly. This states that the `tmpNodes` and `tmpLinks` lists and the `net` object must not be null before the method starts. Furthermore, every element in the `tmpNodes` and `tmpLinks` lists must also not be null, ensuring that all necessary data is present and valid for the network creation process. The `ensures` clause defines the postconditions that must hold true after the method execution completes, assuming the preconditions (specified by `requires`) were true when the method was called. This example specifies that the size of `nNodes` should equal the size of `tmpNodes` and, similarly, for `nLinks` and `tmpLinks`, ensuring that all nodes and links are transferred to the simulation network. It also ensures that every node and link in the simulation network has properly initialized `id` and `simulation status`, which are crucial for the WDS simulation. Regarding the compromised behavior, the `requires` clause includes a condition where if any manipulation is detected in either the nodes or links list, a specific threat (`WDS_TAMPERING`) is implied. The `ensure` clauses look for signs of compromise by checking if any node in the network lacks a valid identifier after network creation (it could indicate a failure in the node initialization process or data corruption) and by checking if any link in the network has not been assigned a simulation status, which is critical for correct simulation operations and could suggest either an omission in processing or a deliberate manipulation. The `alarms` clause specifies that an alarm indicating the threat should be activated if the manipulation detection function returns true for either list. We should note that this example requires the definition of a helper method (`isManipulated(List<?> items)`) that checks whether the list of nodes or links has any anomalies that suggest WDS data tampering or integrity issues.

Listing 2. Example of security specification for creating a WDS network

```
normal_behavior
    requires  tmpNodes != null && tmpLinks != null
        && net != null;
    requires  \forall int i; 0 <= i && i < tmpNodes.
        size(); tmpNodes.get(i) != null;
    requires  \forall int i; 0 <= i && i < tmpLinks.
        size(); tmpLinks.get(i) != null;
    ensures nNodes.size() == tmpNodes.size();
    ensures nLinks.size() == tmpLinks.size();
    ensures \forall int i; 0 <= i && i < nNodes.size
        (); nNodes.get(i).getId() != null;
    ensures \forall int i; 0 <= i && i < nLinks.size
        (); nLinks.get(i).getSimStatus() != null;

compomised_behavior
    requires (isManipulated(tmpNodes) ||
        isManipulated(tmpLinks)) ==> Threat.
        WDS_TAMPERING;
    ensures \exists int i; 0 <= i && i < nNodes.size
        (); nNodes.get(i).getId() == null;
    ensures \exists int i; 0 <= i && i < nLinks.size
        (); nLinks.get(i).getSimStatus() == null;
    alarms (Threat.WDS_TAMPERING) isManipulated(
        tmpNodes) || isManipulated(tmpLinks);

protected void createSimulationNetwork(List<Node>
    tmpNodes, List<Link> tmpLinks, Network net) {
    ...
}
```

We can apply the security monitoring approach to WDS network components as well. Listing 3 provides an example of specifying normal and compromised behavior for the `setDiameterAndUpdate` method, which defines and updates the diameters of WDS nodes. Regarding the normal behavior, the `requires` clause ensures that the input diameter is within a realistic range and that the WDS network object is not null; the `ensures` clause verifies that the diameter has indeed been updated and that the change does not exceed a threshold (e.g., 50% of the original value), helping to enforce stability and predictability in WDS modifications. Regarding the compromised behavior, the `requires` clause asserts that the network context parameter must not be null for the method to execute and that if the diameter is found to be manipulated (as determined by the `isDiameterManipulated` helper function comparing the new and old diameter), it should imply a specific type of threat, labeled `DIAMETER_TAMPERING`. The `ensures` clause confirms that the method acknowledges the manipulation (if any) of the diameter, validating the integrity checks implemented within the method. The `alarms` clause declares that the method should raise an alarm of a specific threat (i.e., `DIAMETER_TAMPERING`) if the diameter has been manipulated.

Listing 3. Example of security specification for creating a WDS node

```
normal_behavior
    requires diameter > 0 && diameter < 100;
    requires  net != null;
    ensures \old(diameter) != diameter;
    ensures \old(km) != km;
    ensures ensures Math.abs((diameter - \old(
        diameter)) / \old(diameter)) * 100 <= 50;

compomised_behavior
    requires net != null;
    requires isDiameterManipulated(diameter, \old(
        diameter)) ==> Threat.DIAMETER_TAMPERING;
    ensures isDiameterManipulated(diameter, \old(
        diameter));
    alarms (Threat.DIAMETER_TAMPERING)
        isDiameterManipulated(diameter, \old(
        diameter))

public void setDiameterAndUpdate(double diameter,
    org.addition.epanet.network.Network net) {
    ...
}
```

Moreover, we can apply the security monitoring approach for specific security properties, like deadlocks. Listing 4 provides an example of specifying normal and compromised behavior for the `updateHydraulicStates` method, which updates states based on WDS network conditions and hydraulic computations. Regarding the normal behavior, the `requires` clause ensures that all nodes and links start with valid hydraulic states (pressure for nodes, flow rates for links) within expected ranges; the `ensures` clause guarantees that the hydraulic states (pressure and flow rates) remain unchanged after the execution, indicating no side effects from the method under normal conditions. Regarding the compromised behavior, the `requires` clause checks whether a deadlock condition exists based on the hydraulic states of nodes and links; the `ensures` verifies that even under compromised conditions, the hydraulic states do not change, reflecting the system's inability to proceed due to deadlock; the `alarms` clause specifies that if a deadlock is detected, it should raise an alarm categorized under a specific threat (i.e., `HYDRAULIC_DEADLOCK`).

Listing 4. Example of security specification for deadlocks

```
normal_behavior
    requires (\forall int i; i >= 0 && i < nNodes.
        size(); nNodes.get(i).getPressure() >=
        MIN_PRESSURE && nNodes.get(i).getPressure()
        <= MAX_PRESSURE);
    requires  (\forall int i; i >= 0 && i < nLinks.
        size(); nLinks.get(i).getFlowRate() >=
        MIN_FLOW && nLinks.get(i).getFlowRate() <=
        MAX_FLOW);
    ensures (\forall int i; i >= 0 && i < nNodes.
        size(); nNodes.get(i).getPressure() == \old(
        nNodes.get(i).getPressure()));
    ensures (\forall int i; i >= 0 && i < nLinks.
        size(); nLinks.get(i).getFlowRate() == \old(
        nLinks.get(i).getFlowRate()));

compomised_behavior
    requires isDeadlockCondition(nNodes, nLinks) ==>
         Threat.HYDRAULIC_DEADLOCK;
    ensures (\forall int i; i >= 0 && i < nNodes.
        size(); nNodes.get(i).getPressure() == \old(
        nNodes.get(i).getPressure()));
    ensures (\forall int i; i >= 0 && i < nLinks.
        size(); nLinks.get(i).getFlowRate() == \old(
        nLinks.get(i).getFlowRate()));
    alarms (Threat.HYDRAULIC_DEADLOCK)
        isDeadlockCondition(nNodes, nLinks);

public void updateHydraulicStates() {
    ...
}

protected boolean checkForDeadlocks() throws
    DeadlockDetectedException {
    if (isDeadlockCondition(nNodes, nLinks)) {
        throw new DeadlockDetectedException('
            Deadlock␣detected␣in␣hydraulic␣
            simulation');
    }
    return true;
}
```

The goal of integrating EPANET and ASM2S tools is to synthesize the security monitors automatically (e.g., for the Listings presented). Several steps are required in this direction: i) define formal specifications using JML annotations, expressing `normal_behavior` and `compromised_behavior`, and defining preconditions (`requires`), postconditions (`ensures`), and `alarms`, ii) preprocess to parse the Java source code and identify JML annotations, iii) develop a code generator that translates JML annotations into assertion code, iv)integrate the generated assertion code into the original source code (injection of precondition checks at the beginning of the methods, postcondition checks at the end of the methods, and alarm-raising code where necessary), v) compile and validate the integrated code, vi) deploy the finalized code in WDS for run-time monitoring, and vii) implement a logging mechanism to record violations and an alerting system to notify administrators of potential security breaches.

V. Discussion, Challenges, and Future Work

EPANET and ASM2S tools are used to simulate WDS; they introduce different but complementary capabilities, with the comparative analysis presented in this paper highlighting their strengths and limitations in addressing different challenges of WDS. EPANET stands out for its robust hydraulic and water quality modeling capabilities. On the other hand, ASM2S WDS specializes in security modeling, offering real-time monitoring and detection of cyber threats to ensure the safe and secure WDS operation. Regarding security monitoring, a key feature of critical infrastructure like WDS is that it depends on the expressiveness of the models used by the WDS tools. Considering that EPANET does not support security and resilience WDS requirements, it fails to monitor the security of WDS on the one hand and fails to recover the system (i.e., resilience) from cyber-attacks on the other hand. Consequently, such tools cannot ensure continuous operation of the critical infrastructures in the face of attack, a key requirement of such real-time critical infrastructures. In principle, ASM2S mainly supports the modeling and monitoring of cyber components of the WDS, while EPANET supports the modeling and monitoring of physical components of the critical infrastructure. Consequently, integrating EPANET and ASM2S WDS ensures domain-specific protection for WDS. Considering their cascaded effects, the integration will enable us to detect cyber, physical, and cyber-physical threats to WDS.

Our integrated approach for implementing run-time security monitoring leverages Java programs with JML annotations to specify security properties. This method allows for generating run-time assertions to monitor program behavior and detect potential security threats. While initially tailored to Java, the approach could be extended to support other programming languages. For instance, modifications could be made to adapt the approach to languages like Python, utilizing run-time introspection and reflection mechanisms to generate run-time assertions. However, applying this approach requires understanding the WDS behavior and the security requirements, and proficiency in using JML for annotation. The effort and time required for code annotation may vary depending on factors such as familiarity with JML, the size and complexity of the codebase, developers' skills and experience, and available

tools and support. Prioritizing critical security properties and focusing on essential parts of the codebase can help manage the annotation process effectively.

A primary objective of such integrations is the automatic synthesis of security monitors in WDS. While there are steps that we can take to implement such approaches, as discussed in the previous section, the feasibility of an automatic synthesis depends on several factors: i) the synthesis requires extensive formal specifications that capture all normal and compromised behaviors, ii) the performance overhead must be minimal to avoid impacting the WDS operational efficiency, and iii) the synthesis requires thorough testing and validation to ensure the generated monitors correctly identify and respond to security threats without causing false alarms.

Future work should focus on fine-tuning and extending the integrated EPANET-ASM2S solution. Specifically, the exploitation of advanced security monitoring features, such as anomaly detection algorithms and adaptive response mechanisms, will be highly interesting in facing emerging cybersecurity challenges of distributed ICS environments. In addition, to speed up the adoption of integrated WDS management solutions across the entire industry, standardizing and streamlining the integration process will be one of the future research directions.

VI. Conclusion

Water distribution systems are a crucial domain in the Industrial Control Systems landscape. However, the increasing interconnectivity and the distributed nature of such systems bring risks to their cybersecurity. Many well-established WDS software tools, like EPANET, still need to be equipped with features to tackle such risks. However, other solutions, such as ASM2S, which can complement the tools mentioned above well, could meet these objectives. In this paper, we analyzed the capabilities offered by EPANET and ASM2S tools and took the first steps in exploring their integration, leading to water distribution systems with enhanced hydraulic, water quality, and cybersecurity modeling characteristics.

Acknowledgments

The research work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the 2nd Call for HFRI's Research Projects to Support Faculty Members & Researchers (Project Number: 4012).

References

[1] G. E. Raptis, M. T. Khan, K. Stefanidis, C. Koulamas, and D. Serpanos, "Towards run-time security monitoring of distributed industrial control systems," in *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sep. 2023. [Online]. Available: http://dx.doi.org/10.1109/ETFA54631.2023.10275618

[2] E. D. Knapp and J. T. Langill, *Security monitoring of industrial control systems*, 2nd ed. Boston: Syngress, 2015, pp. 351–386. [Online]. Available: https://doi.org/10.1016/B978-0-12-420114-9.00012-5

[3] M. Lyu, H. H. Gharakheili, and V. Sivaraman, "A survey on enterprise network security: Asset behavioral monitoring and distributed attack detection," *IEEE Access*, pp. 89 363–89 383, 2024. [Online]. Available: http://dx.doi.org/10.1109/ACCESS.2024.3419068

[4] N. Moustafa, K.-K. R. Choo, and A. M. Abu-Mahfouz, "AI-enabled threat intelligence and hunting microservices for distributed industrial IoT system," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1892–1895, 2021. [Online]. Available: https://doi.org/10.1109/TII.2021.3111028

[5] M. A. Poltavtseva, "Heterogeneous data aggregation and normalization in information security monitoring and intrusion detection systems of large-scale industrial CPS," *Proceedings of the Institute for System Programming of the RAS*, vol. 32, no. 5, pp. 131–142, 2020. [Online]. Available: https://doi.org/10.15514/ispras-2020-32(5)-10

[6] T. Cruz, J. Barrigas, J. Proença, A. Graziano, S. Panzieri, L. Lev, and P. Simões, "Improving network security monitoring for industrial control systems," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 878–881. [Online]. Available: https://doi.org/10.1109/inm.2015.7140399

[7] M. Yan, H. Huang, and J. Li, "Research on key technologies of industrial internet data security," *Journal of Physics: Conference Series*, vol. 1883, no. 1, pp. 1–8, Apr. 2021. [Online]. Available: https://doi.org/10.1088/1742-6596/1883/1/012087

[8] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Ssemantic security monitoring for industrial processes," in *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, Dec. 2014. [Online]. Available: https://doi.org/10.1145/2664243.2664277

[9] M. T. Khan, D. Serpanos, and H. Shrobe, "ARMET: Behavior-based secure and resilient industrial control systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 129–143, 2018. [Online]. Available: https://doi.org/10.1109/JPROC.2017.2725642

[10] M. T. Khan and I. Tomic, "Securing industrial cyber-physical systems: A run-time multilayer monitoring," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6251–6259, Sep. 2021. [Online]. Available: https://doi.org/10.1109/tii.2020.3032968

[11] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, and A. Ostfeld, "Simulation of cyber-physical attacks on water distribution systems with EPANET," in *Proceedings of the Singapore Cyber-Security Conference (SGCRC)*. IOS Press, 2016, pp. 123–130. [Online]. Available: https://doi.org/10.3233/978-1-61499-617-0-123

[12] C. Karrenberg, J. Benavides, E. Berglund, E. Kang, and J. Baugh, "Identifying cyber-physical vulnerabilities of water distribution systems using finite state processes," in *2nd International Joint Conference on Water Distribution System Analysis and Computing and Control in the Water Industry, WDSA CCWI*, Jul. 2022.

[13] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, and A. Ostfeld, "Characterizing cyber-physical attacks on water distribution systems," *Journal of Water Resources Planning and Management*, vol. 143, no. 5, pp. 04 017 009:1–12, Feb. 2017. [Online]. Available: https://doi.org/10.1061/(ASCE)WR.1943-5452.0000749

[14] H. H. Mahmoud, W. Wu, and Y. Wang, "WDSchain: A toolbox for enhancing the security using blockchain technology in water distribution system," *Water*, vol. 13, no. 14, pp. 1944:1–18, Jul. 2021. [Online]. Available: http://dx.doi.org/10.3390/w13141944

[15] Q. Lin, S. Verwer, R. Kooij, and A. Mathur, *Using datasets from industrial control systems for cyber security research and education*. Springer International Publishing, Dec. 2019, pp. 122–133. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-37670-3_10

[16] U. Parajuli and S. Shin, "Identifying failure types in cyber-physical water distribution networks using machine learning models," *Aqua*, vol. 73, pp. 504–519, Feb. 2024. [Online]. Available: https://doi.org/10.2166/aqua.2024.264

[17] C. Sewell, H. Mehrpouyan, S. O'Toole, E. Moseley, and S. Reese, "Cybersecurity of critical infrastructure in water distribution systems," in *ICUR Proceedings*, 2021.

[18] S. O'Toole, C. Sewell, and H. Mehrpouyan, "IoT security and safety testing toolkits for water distribution systems," in *2021 8th International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, Dec. 2021, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/IOTSMS53705.2021.9704886

[19] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy Magazine*, vol. 9, no. 3, pp. 49–51, May 2011. [Online]. Available: http://dx.doi.org/10.1109/MSP.2011.67