http://dx.doi.org/10.62762/TC.2025.281528

RESEARCH ARTICLE

USVScout: Detecting Security Vulnerabilities in USV-based Marine Applications

Shihao Zhou[1,*]

[1] Three Gorges Navigation Authority, Yichang, China

Abstract

With the deep integration of artificial intelligence (AI) and Internet of Things (IoT) technologies, the maritime transportation industry is undergoing profound changes, and the application scenarios of unmanned surface vehicles (USVs) are constantly expanding. Aiming at the security threats faced by USV application systems, this paper proposes a new threat model for ship communication protocols and service logics, and the system covers the potential adversarial entity attack paths in application program logics and communication protocols. Based on this model, the automated security inspection framework USVScout was designed and implemented. By parsing the source code of the application program, the security analysis was formalized as an attribute verification task under the standard computing model, significantly improving the systematicness and accuracy of the detection. The experimental results show that in three real Marine application scenarios, USVScout successfully identified new types of vulnerabilities that static analysis tools failed to detect, and it can achieve sub-second real-time detection of 12 types of vulnerability patterns on an ordinary laptop, verifying the significant advantages of the method in terms of security and efficiency. The threat modeling and automated detection framework proposed in this paper provides technical support for ship network attack and defense drills and lays the foundation for building an intelligent and secure maritime transportation system.

Keywords: USVScout, threat modeling, automated security inspection, maritime transportation.

1 Introduction

Unmanned surface vehicles (USVs) are increasingly widely used in the Marine industry. Unlike traditional manned vessels, USVs can be operated through remote control or with a certain degree of autonomous decision-making ability, thereby effectively reducing operational risks and costs. In recent years, with the rapid development of artificial intelligence (AI) and Internet of Things (IoT) technologies, the application scope of USV has been significantly expanded, covering many tasks that were previously difficult to achieve. For instance, USV has been applied in fields such as Marine rescue [1], bathymetry and photogrammetry [2], and algal bloom removal [3]. Although some USV applications have entered the practical usage stage, a considerable number are still in the prototype development stage. However, a stable communication network is fundamental to all USV systems, regardless of their maturity level, both to achieve remote control or status monitoring. This is

Submitted: 21 June 2025
Accepted: 10 July 2025
Published: 01 September 2025

Vol. 1, **No.** 1, 2025.
10.62762/TC.2025.281528

***Corresponding author:**
Shihao Zhou
2772613724@qq.com

Citation
Zhou, S. (2025). USVScout: Detecting Security Vulnerabilities in USV-based Marine Applications. *ICCK Transactions on Cybersecurity*, 1(1), 17–34.

© 2025 ICCK (Institute of Central Computation and Knowledge)

because both command execution and autonomous decision-making depend on reliable data transmission to ensure safe and efficient operation.

Integrating unmanned aerial vehicles (UAVs) with unmanned surface vehicles (USVs) can significantly enhance their collaborative mission execution capabilities. UAV is capable of identifying distressed vessels or environmental threats in a large sea area and sending control instructions to USV. Meanwhile, the USV can also transmit the sensor data back to the UAV in real time for analysis and processing. This air-sea collaborative architecture effectively reduces the reliance on manned vessels and enhances the efficiency and sustainability of Marine operations [4].However, the high degree of interconnection of the system has also brought new cyber security risks. With the increasingly close communication and cooperation between UAV and USV, the exposed attack surface continues to expand, which significantly increases the possibility of network intrusion.

To effectively prevent such cyber attacks, multi-level security protection mechanisms must be deployed. Among them, the communication link between the UAV and the USV is a key security weakness that urgently needs focused protection. Although the Internet of things (IoT) is commonly used in the field of security mechanism, such as access control and encryption technology provides the foundation to a certain extent safeguard [5, 6], but it is still difficult to achieve in the complex sea environment real end-to-end security [5, 6]. In addition, the common communication protocol (such as the MQTT) due to the improper implementation defects or configuration in the actual deployment and introducing the safe hidden trouble, as potential attacks entry [7, 8]. What is more alarming is that untrusted third-party devices accessed through the USV-UAV interface may be used to transmit malicious instructions, thereby undermining the integrity of the system. With the continuous improvement of the integration degree of UAV and USV systems, it may also be necessary to introduce new security mechanisms, which are usually beyond the technical capabilities of developers lacking a security background.

In recent years, some studies have focused on the safety issues of maritime transportation systems. For a review of the relevant work, please refer to Section section 3 for a detailed discussion. However, the existing research mainly focuses on the security analysis of the underlying communication protocol and has not yet covered the composite application scenarios that integrate UAVs and USVs simultaneously. More importantly, these studies generally do not explore the security characteristics of the actual systems constructed based on the USV communication protocol at the application level. Therefore, the current security analysis tools and protection mechanisms designed for Internet of Things (IoT) devices may not be directly applicable to the special environment of maritime transportation.

In order to understand the security issues in practical applications, it is necessary to study the generation mechanism of vulnerabilities and their detection methods. This paper proposes a novel threat model for applications based on Unmanned Surface Vehicles (USVs). In this model, attackers can simulate the roles of remote users, on-board users, and on-board users accessed through unmanned aerial vehicles (UAVs), and thereby launch attacks with the ability to impersonate any of the above identities. Based on this threat model, a security inspection framework named USVScout is designed. USVScout takes the source code of untrusted applications as input and automatically verifies whether they violate critical security attributes. The framework is grounded in standard computing model theory and transforms the security check problem into a security attribute verification task. USVScout extracts the state transition system from the source code with the help of the transition technique proposed in the literature [9] and applies the NeS tools [10] to verify the system's security properties, thereby enabling safety analysis of the application.

To evaluate the security analysis capabilities of USVScout in practical applications, the framework was applied to three real-world Marine applications. The experimental results show that in all the tested applications, USVScout successfully identified several new types of vulnerabilities. These vulnerabilities are exploitable under the threat model proposed in this paper.And it has not been discovered by existing mainstream static analysis systems [11–14]. It is worth noting that in one of the applications, USVScout did not detect any security violations, indicating that the program has security guarantees under the current threat model.All experiments were run on a regular laptop. USVScout was able to complete the security attribute verification of each application and output the analysis results within seconds, demonstrating excellent performance efficiency.

The contributions of our work are:

1. A new threat model for USV-based applications is introduced.
2. An automated security inspection framework for USV applications, named USVScout, is designed and implemented.
3. The application of USVScout to marine applications is demonstrated.

The threat model and security inspection framework proposed in this paper are not only applicable to marine applications, but also can be extended to other application scenarios based on USVs. At present, many systems in maritime transportation have deployed USVs. This research is expected to contribute to enhancing the security and reliability of such applications.

2 Unmanned Surface Vehicles

This section provides a brief introduction to Unmanned Surface Vehicles (USVs), and it is assumed that the reader has a basic understanding of the fundamental concepts involved.

The Unmanned Surface Vehicle (USV) is a mobile platform that can operate on the water surface through autonomous decision-making or remote control without the presence of personnel. Its structural form is not limited to the traditional ship form, but can also be flexibly designed into diverse structures according to the task requirements. A typical USV system is usually equipped with a sensor array and actuators for environmental perception and task execution. Some systems integrate local processing units (such as embedded systems or single-board computers) to support real-time data processing; In addition, some USVs can perform collaborative operations with remote processing units to achieve distributed computing capabilities. Most USVs are also equipped with wireless communication modules (such as Wi-Fi or cellular networks) to enable data interaction with the control center or other devices.

The energy supply mode of USV is closely related to its application scenarios. Prototype systems are mostly powered by batteries, while USVs in commercial applications may rely on fossil fuels, solar energy or hybrid power solutions. The range varies depending on the energy source: for example, battery-powered USVs typically operate for 4-8 hours, while fuel-powered USVs can operate around the clock.

This paper focuses on an unmanned surface vehicle (USV) equipped with an onboard processing unit (PU) and capable of communicating with a remote processing unit. The remote PU can interact with the USV through the Web user interface (Web UI) or the command-line interface (CLI). In addition, USV may also regularly send status reports to the remote PU through the Message Queue (MQTT) protocol to achieve remote monitoring.

Unmanned aerial vehicles (UAVs) can essentially be regarded as a type of USV, but they are usually not designed for commercial applications and their operating costs are generally higher than those of traditional USVs. Uavs are usually powered by batteries and are subject to the relevant limitations of their flight areas. They are also equipped with on-board processing units and may integrate wireless communication modules to enable data interaction with remote processing units.

Unmanned Surface Vehicles (USVs) have been widely applied in multiple fields. This article focuses on three typical application scenarios where USVs are integrated into maritime transportation systems. The application cases studied have the following characteristics: (1) They have been deployed and operated in the actual system, or (2) They are still in the prototype development stage at present; However, (3) As the related hardware and software components have become mature, they possess good scalability and rapid deployment potential.

One application is maritime rescue (see Figure 1). Suppose a ship crew sends out a mayday signal. The remote processing unit (remote PU) can coordinate and dispatch one or more unmanned aerial vehicles (UAVs) and unmanned surface vehicles (USVs) to carry out survivor search and rescue tasks simultaneously in multiple target areas. The USV can receive movement instructions issued by the remote PU through the Command Line interface (CLI) and drive to the designated location accordingly. Meanwhile, it can also periodically synchronize the status with the remote PU through the Message Queue (MQTT) protocol: on the one hand, it receives query requests from the remote PU; on the other hand, it actively reports the current location information and the ship targets detected in this area. When the remote PU collects a sufficient number of status reports, it can determine whether the search and rescue target of the

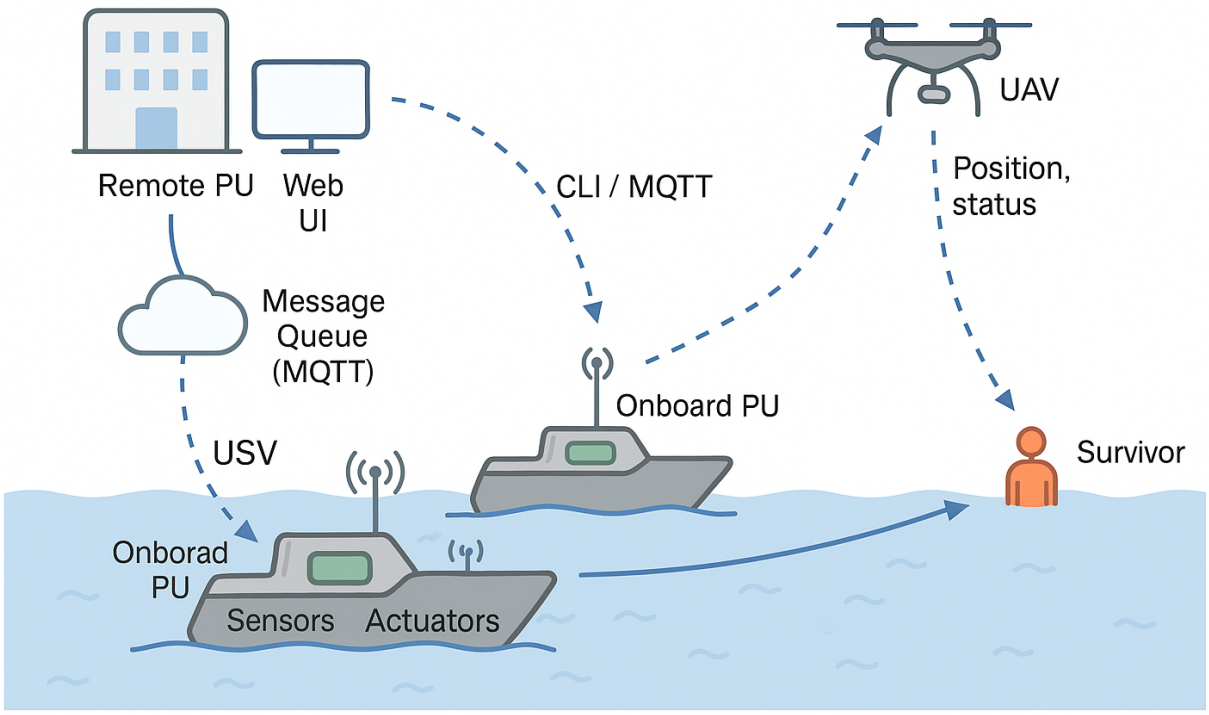

Figure 1. USV-based maritime rescue system.

current area has been completed based on the progress of the task, and issue instructions again through the CLI to allocate the USV to the next task area to continue the task execution.

Another typical application involves bathymetric tasks (used to study water depth) and photogrammetric tasks (for three-dimensional terrain reconstruction based on images) (see Figure 2). Among these, bathymetric technology can be employed to construct submarine topography models, which are of great significance in scenarios such as navigation path planning, geological disaster monitoring, and emergency disaster relief. Photogrammetry, on the other hand, can simultaneously capture geomorphic information of both water areas and land, thereby providing support for the fusion of multi-source geospatial data. In such application scenarios, the remote processing unit (remote PU) can collaboratively schedule an unmanned aerial vehicle (UAV) and an unmanned surface vehicle (USV) to achieve integrated air–sea task execution. The USV receives navigation instructions issued by the remote PU through the Command Line Interface (CLI) and then navigates to the designated area accordingly. Upon arrival, the USV utilizes its GPS module, radar system, and cameras to collect positioning information, map the coastline, and capture high-resolution image data. Subsequently, the USV transmits the collected data back to the remote PU in real time via its wireless communication module (such as Wi-Fi or a cellular network) for further analysis and processing.

The third typical application is waterway maintenance (see Figure 3). The sediment carried by the water flow is prone to deposit in the waterway, affecting the navigable capacity. In severe cases, it may even cause navigation safety accidents. Regular dredging can effectively maintain the water depth of the waterway and ensure the safety and smoothness of water traffic. In such tasks, the remote processing unit (remote PU) can collaboratively schedule unmanned aerial vehicles (UAVs) and an unmanned surface vehicle (USV) to achieve integrated air-sea dredging operations. The USV can receive navigation instructions issued by the remote PU through the Command Line interface (CLI), autonomously drive to the target area, and use the actuators it is equipped with (such as buckets or suction devices) to clear the sediment. Meanwhile, the USV can also communicate periodically with the remote PU through the Message Queue (MQTT) protocol, receive control instructions and report the current dredging status, thereby achieving remote monitoring and dynamic management of the entire operation process.

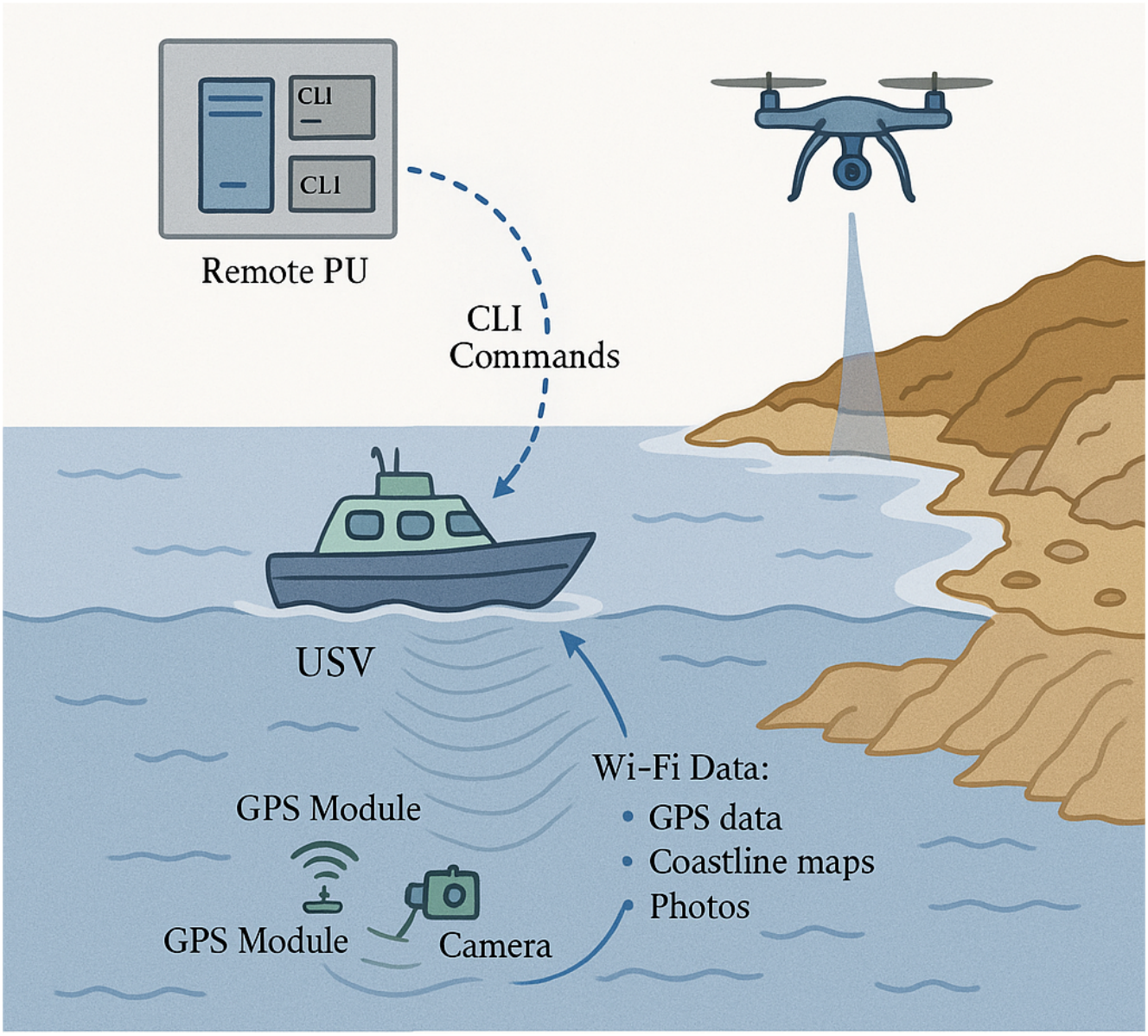

Figure 2. USV-based bathymetric and photogrammetric measurements.

3 Related Works

In recent years, in view of the network attack risk increasing ship systems, frequent related security incidents, highlights the maritime communication and control system security challenge [15, 16]. Existing studies have shown that in the communication scenarios between Vessel to Vessel (V2V) and Vessel to Shore (V2S), the current shore-based communication mechanism has significant security risks, mainly due to its extensive use of algorithms with weak encryption strength. Vulnerable to threats such as man-in-the-middle attacks [17]. In addition, the research on Global Navigation Satellite System (GNSS) pointed out that the positioning information lacks sufficient protection mechanism, and it is vulnerable to GPS spoofing and jamming attacks, which will lead to navigation path deviation and even navigation failure [18]. In ship operating system (OS) level, the research also reveals multiple critical systems there are serious loopholes, part can be utilized remotely to gain root access, and complete control of a target device [19, 20]. Furthermore, research on the communication mechanism of unmanned surface vehicles (USVs) indicates that many current USVs still rely on unencrypted or less secure protocols, such as short baseline navigation protocols, whose data is vulnerable to tampering or forgery. There is higher risk of spoofing attacks [19, 20]. Based on the above research results, this paper focuses on the modeling and verification of security attributes of USV-based applications, proposes an automated security inspection framework - USVScout, and applies

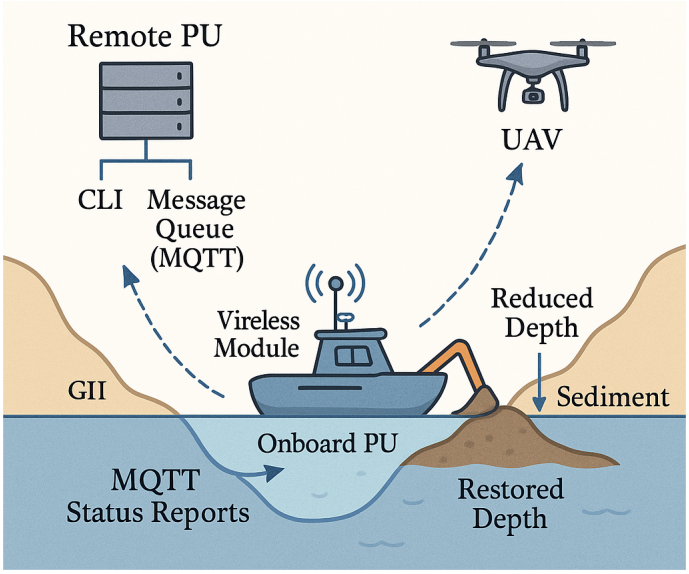

Figure 3. USV-based waterway maintenance.

it to real-world Marine application systems to evaluate their security performance under actual threat models.

Existing work on USV security focuses on the protocol level. The work in [21] uses formal methods to study the security of V2V protocols. The work in [22] proposes a security checking framework for maritime transport. The works in [23–25] propose USV navigation systems. The works in [26–28] propose systems for time, identity, and communication security. This paper is also committed to enhancing the safety of applications based on unmanned surface vehicles (USVs), but adopts a different technical approach from existing research. The focus is on integrating USVs into practical applications in maritime transportation systems and automatically verifying their safety and security attributes using source code as input. The proposed method is based on the computational standard model theory. By reducing the verification problem of safety attributes to the safety verification problem, it realizes the formal analysis and inspection of the system's safety characteristics.In contrast, the method in the literature [29] does not achieve the automatic verification of security attributes;The literature [30] does not focus on practical applications in the real world.And literature [26, 27, 31, 32] Although the proposed method involves communication security, it has not been studied for applications that use the USV communication mechanism.

There is related work on the static analysis of IoT applications. The research in [11] proposes a system, Soteria, that uses model checking to check safety and security properties of IoT applications. The research in [12] proposes a system, iRuler, that uses SMT solving to check inter-rule policy vulnerabilities in trigger-action platforms. The research in [13] proposes a system, IoTGuard, that uses dynamic enforcement to check safety and security policy violations at runtime. The research in [33] proposes a system, TCPwn, that uses model checking to find attacks against TCP congestion control implementations. The research in [14] proposes a system, SmartPatch, that uses automated testing and patch generation to correct vulnerabilities in IoT platforms. The state-of-the-art approach in [10] is used for safety verification in the security checkers. The approaches in [11, 12] have lower verification times and are specific to trigger-action platforms. The approaches in [13, 33] have low precision and are not specific to USV-based applications. The approach in [14] has a lower verification time, but is not designed for safety property verification. A summary comparison of these related works and our approach is shown in Table 1.

4 Threat Model

In this section, a threat model for Unmanned Surface Vehicle-based (USV) applications is proposed. This model introduces three shore-based attackers to describe the behavioral characteristics and attack capabilities that potential attackers may possess. Each role model describes the identities that an attacker may assume in the system and lists the typical attack types that can be executed.

The role model used is based on the attack modeling framework of Internet of Things (IoT) platform proposed in [14]. This study has verified the effectiveness of related attacks on real platforms, and shows that the model is suitable for the current mainstream IoT application scenarios. On this basis, this paper extends and adapts the original model to reflect more comprehensively the security threats that USV-based applications may face in actual deployment.

4.1 Remote Attacker

This role model describes potential attackers who attack applications through the remote PU. Such attackers may already have access to the remote PU, for example, by exploiting vulnerabilities in its operating system or deploying malware, etc., to achieve intrusion.

As shown in Table 2, two roles that potential attackers may assume are identified: (1) legitimate remote PU user, and (2) onboard user. Attackers may also

Table 1. Comparison of related work and our approach.

Work / System	Domain	Focus Area	Level of Analysis	Real Apps	Automated Sec Check	Technique
Tullsen et al. [21]	V2V	Protocol Security	Protocol Level	×	×	Formal Methods
Grigoriánis et al. [22]	Maritime	Security Framework	System Level	×	×	Framework Design
Zhang et al. [23], Sun et al. [24], Akram et al. [25]	USV Navigation	Navigation Security	Component Level	×	×	Control Algorithms
Boretti et al. [26], Berbecam et al. [27]	GNSS/Comm	Identity, Time, Comm	Protocol Level	×	×	Custom Protocol Design
Soteria (Celik et al. [11])	IoT	Safety + Security	Application Logic	✓	✓	Model Checking
iRuler (Wang et al. [12])	IoT (Trigger-Action)	Policy Conflicts	Rule-based Logic	✓	✓	SMT Solving
IoTGuard (Celik et al. [13])	IoT	Runtime Enforcement	Runtime	✓	×	Dynamic Monitoring
SmartPatch (Yuan et al. [14])	IoT	Auto Patching	Patch Level	✓	×	Auto Testing/Repair
This Work (Ours)	USV Applications	App-level Security	Source Code + Model	✓	✓	Model Checking (Safety)

perform the following attacks: (1) masquerade as a legitimate remote PU user, (2) send commands or messages with a poisoned content, (3) inject commands or messages via other channels (e.g., physical access), (4) send commands or messages at a poisoned timing, or (5) send commands or messages with poisoned semantics or context.

4.2 Onboard Attacker

This persona model describes potential attackers that attack the applications via the onboard PU. Such attackers have physical access to the onboard PU.

As shown in Table 3, three roles that potential attackers may assume are identified: (1) legitimate onboard PU user, (2) legitimate remote PU user, and (3) onboard PU user. Attackers may also perform the following attacks if they take on the (1) onboard PU user role: (1) masquerade as a legitimate onboard PU user, (2) privilege escalation attack to gain more privilege, (3) send commands or messages with poisoned semantics or context. Attackers may also perform the following attacks if they take on the (2) remote PU user role: (1) masquerade as a legitimate remote PU user, (2) inject commands or messages via other channels (e.g., physical access), (3) send commands or messages at a poisoned timing, (4) send commands or messages with poisoned semantics or context.

In typical application scenarios such as maritime rescue and channel maintenance sounding, shipborne attackers or shoreline attackers with physical access conditions may all have the behavioral capabilities of long-range attackers. For example, in the maritime rescue system, since the remote processing unit (remote PU) is usually deployed in the ship control room, attackers can launch a remote form of intrusion through physical contact devices; In the channel maintenance system, shore-based attackers can also enter the onboard processing unit (PU) in a similar way to carry out long-range attacks. After completing the extended mission, attackers may also access the processing unit of the USV when it returns to the mothership or maintenance site, and then carry out a long-range attack mode. However, in terms of the feasibility of actual attacks, remote attackers pose a more realistic threat compared to shipborne attackers. This is because physical access to shipborne processing units usually faces higher thresholds, while remote attacks are more likely to achieve penetration and control through network channels.

Table 2. Persona model of remote attackers.

Role	
Legitimate role	Remote PU user
Attack role	Remote PU user
Attack Types	
Attack role as Remote PU user	
(1) Impersonation attack	Masquerade as legitimate remote PU user to launch attacks
(2) Data poisoning attack	Send commands/messages with poisoned content
(3) Data injection attack	Inject commands/messages via alternative channels (e.g., physical access)
(4) Timing attack	Send commands/messages at manipulated timing
(5) Logic manipulation attack (semantic)	Send commands/messages with poisoned semantics
(6) Logic manipulation attack (contextual)	Send commands/messages with poisoned context

Table 3. The persona model of onboard attackers.

Role	
Legitimate role	Onboard PU user
Attack role	Onboard PU user; Remote PU user
Attack Types	
Attack role as Onboard PU user	
(1) Impersonation attack	Masquerade as legitimate onboard PU user to launch attacks.
(2) Privilege escalation attack	Escalate onboard PU user's access range to gain more privilege.
(3) Logic manipulation attack (semantic)	Send commands or messages with poisoned semantics.
(4) Logic manipulation attack (contextual)	Send commands or messages with poisoned context.
Attack role as Remote PU user	
(1) Impersonation attack	Masquerade as legitimate remote PU user to launch attacks.
(2) Data injection attack	Inject commands/messages via other channels (e.g., physical access).
(3) Timing attack	Send commands or messages at a poisoned timing.
(4) Logic manipulation attack (semantic)	Send commands or messages with poisoned semantics.
(5) Logic manipulation attack (contextual)	Send commands or messages with poisoned context.

4.3 Onboard-with-UAV Attacker

This persona model describes potential attackers that attack the applications via an onboard PU when it is near a UAV.Such attackers may disguise themselves as legitimate shipborne processing unit users to gain unauthorized access to the system and carry out malicious operations.As shown in Table 4, a ship attacker carrying a uav could also launch a USV-UAV landing attack by sending a malicious UAV landing command with incorrect semantics or context. Thereby interfering with or controlling the collaborative operation between the USV and the UAV. In the bathymetry and waterway maintenance systems, if the involved UAV is an Uncrewed Aerial Vehicle, then such attackers can play the role of the user of the onboard PU. In maritime rescue systems, attackers may have such role permissions only when the UAV is a manned aircraft (i.e., the mother ship itself).

In the threat model constructed in this paper, some attack behaviors can theoretically be prevented through access control or identity authentication mechanisms. However, since malicious users may bypass these mechanisms in multiple ways, relying solely on access control and identity authentication

Table 4. The persona model of onboard-with-UAV attackers.

Role	
Legitimate role	No role
Attack role	Onboard PU user
Attack Types	
Attack role as Onboard PU user	
(1) Masquerade attack	Masquerade as legitimate onboard PU user.
(2) UAV landing attack	Send UAV landing commands with poisoned semantics or context.

is insufficient to comprehensively defend against all kinds of attacks covered in the model.Specifically, some attacks (such as sending commands with incorrect semantics) can still be successfully carried out even when the attacker has been authenticated and authorized; Some other attacks (such as injecting commands through unauthorized channels) directly circumvent the access control mechanism; However, attack forms such as time series attacks are completely unable to be effectively defended against through traditional access control or identity authentication methods.

Based on the above analysis, this threat model focuses on those security threats that cannot be prevented through access control or identity authentication mechanisms. Nevertheless, access control and identity authentication remain indispensable basic security measures, and their core role lies in preventing the attack roles defined in the model from obtaining legitimate system identities.For example, in the absence of an effective access control mechanism, attackers may impersonate users of the remote PU and then launch disguise attacks; In the case of well-configured access control, such attacks will be effectively curbed. Therefore, this model does not consider the types of attacks that can be blocked by rationally configuring access control and identity authentication mechanisms.

5 USVScout

5.1 Security Property Preliminaries

It is necessary to systematically study a set of safety attribute spaces closely related to applications based on Unmanned Surface Vehicles (USVs) to support in-depth analysis and formal verification of their safety requirements. To this end, several core safety attributes are proposed for typical marine application scenarios, aiming to more comprehensively depict the safety guarantee mechanisms and behavioral constraints that such systems should possess.

Critical communication. When the USV receives a navigation command from the remote PU, it may be directed toward potentially hazardous regions, such as areas with floating icebergs. Due to the inherent latency in long-range navigation and limitations in real-time command revocation, the USV may be unable to receive updated instructions in time to avoid danger. To prevent potential accidents, it is essential for the USV to incorporate onboard safety mechanisms that allow it to dynamically assess its current trajectory. If a dangerous area is detected along the path, the USV should autonomously divert to a safer route before entering the hazardous zone.

Critical messages. The remote PU may communicate with the USV by sending messages through a message queue. Some of these messages may contain location information, prompting the USV to adjust its current trajectory and navigate to the specified destination. Other messages may request that the USV report sensor data or other information it has collected during its operation. To ensure mission correctness and system responsiveness, it is crucial that the USV reliably receives and processes all incoming messages from the remote PU without omission. Furthermore, the USV must transmit its collected data back to the PU in a timely manner to support real-time monitoring and decision-making.

Power management. The remote PU may have the capability to remotely shut down the USV. However, if the remote PU is compromised, it may issue shutdown commands at inappropriate times, potentially resulting in severe consequences. For example, in bathymetric or photogrammetric survey applications, a premature shutdown may prevent the USV from returning to a safe location before battery depletion. In maritime rescue scenarios, an unauthorized shutdown could disable the USV during

a critical emergency, potentially leading to mission failure or loss of life. An overview of such an attack flow in a maritime rescue context is illustrated in Figure 4. To prevent such risks, the USV must ensure that shutdown commands are executed only under appropriate conditions and originate from a verified, uncompromised remote PU.

Multiple USVs. In some deployment scenarios, the remote PU may be authorized to control multiple USVs simultaneously. However, if the remote PU is compromised, it could issue malicious commands to all controlled USVs, amplifying the impact of an attack. For instance, a compromised PU might direct all USVs to navigate toward a hazardous area or disrupt their coordinated operation.To mitigate such risks, it is necessary to ensure that multi-USV control is exercised only when explicitly required and under secure conditions. Furthermore, each USV should independently validate the content and context of received commands before execution, rather than relying solely on the authority of the issuing PU.

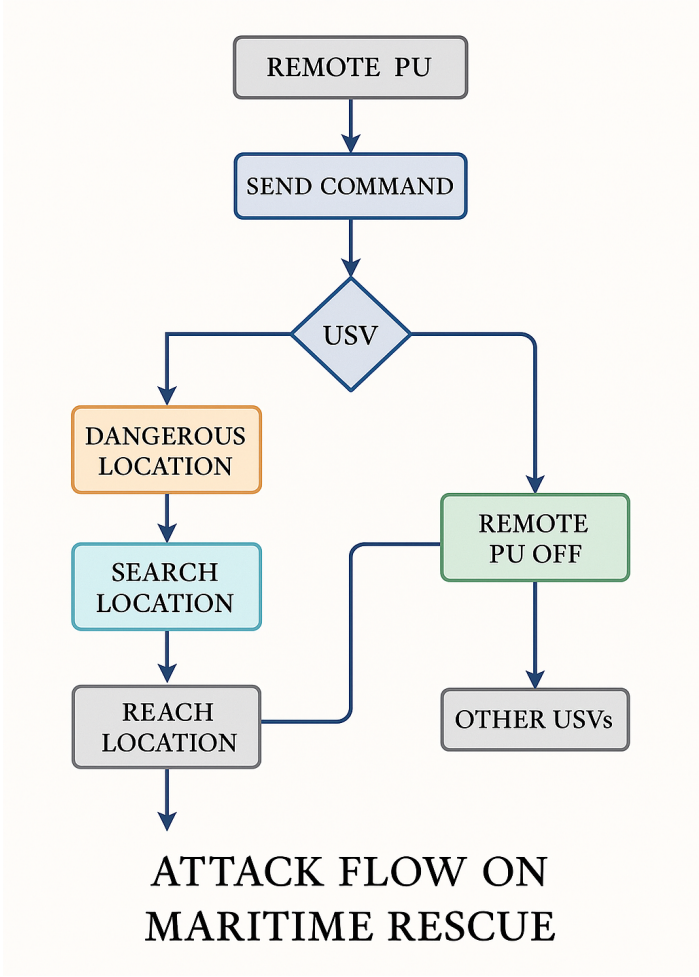

Figure 4. Attack flow on maritime rescue.

5.2 Security Property Specification and Verification

USVScout, a security checking framework for multi-persona USV-based applications, is introduced. USVScout takes the source code of an untrusted application as input and automatically generates safety properties for formal verification. It employs the NeS model checker [10] to verify these properties.To enable model checking, USVScout first translates the application source code into an *abstract state transition system* (ASTTS) using the code-to-model translation technique proposed in [10]. The resulting ASTTS encodes sufficient semantic information to specify and verify safety properties of the form $\{\phi \mid \psi\}$. The USV-based system includes three principal roles: the remote PU, the on-board PU, and the USV. These entities communicate via commands and messages, whose semantics are captured using logical predicates. For example, commands issued by the remote PU to the USV are represented by predicates such as `command_to_USV_navigate_to_location(loc:integer)` or `command_to_USV_move_to_location(loc:integer)`, where loc denotes the intended destination. Messages from the remote PU to the USV, such as status requests, are represented by predicates like `message_from_remote_PU_to_USV_report_status()`. Periodic reports sent from the USV to the remote PU, such as GPS updates, are captured using predicates like `USV_reports_GPS_data()`.(see Figure 5)

Security properties are specified using predicates to form Disjunctive Form Automata (DFA). Formally, a property is specified using a tuple $\langle \mathcal{F}, \mathcal{I}, \mathcal{T}, \mathcal{S}, \mathcal{A} \rangle$, where $\mathcal{F}$ is a set of predicates that specify the failure states, $\mathcal{I}$ is a set of predicates that specify the initial states, $\mathcal{T}$ is a set of transitions between states, $\mathcal{S}$ is a set of states, and $\mathcal{A}$ is a set of labels for states. The property is satisfied if the system starts at an initial state, goes through at most one transition per trace, and ends in a non-failure state. The transitions between states are represented using predicates and labels. A transition from one state to another with label l is denoted as $(\mathcal{S}_1, l, \mathcal{S}_2) \in \mathcal{T}$. If a transition with label l is from one state to another without any predicate, then this transition is denoted as $(\mathcal{S}_1, l, \mathcal{S}_2) \in \mathcal{T}$. If a transition with label l is from one state to another with predicate p, then this transition is denoted as $(\mathcal{S}_1, l, p, \mathcal{S}_2) \in \mathcal{T}$. If there is no transition between two states, then it is denoted as $(\mathcal{S}_1, l, \mathcal{S}_2) \notin \mathcal{T}$. If there is no transition with predicate p, then it is denoted as $(\mathcal{S}_1, l, p, \mathcal{S}_2) \notin \mathcal{T}$.

Examples of properties are shown below:

Maritime rescue Property: Never send

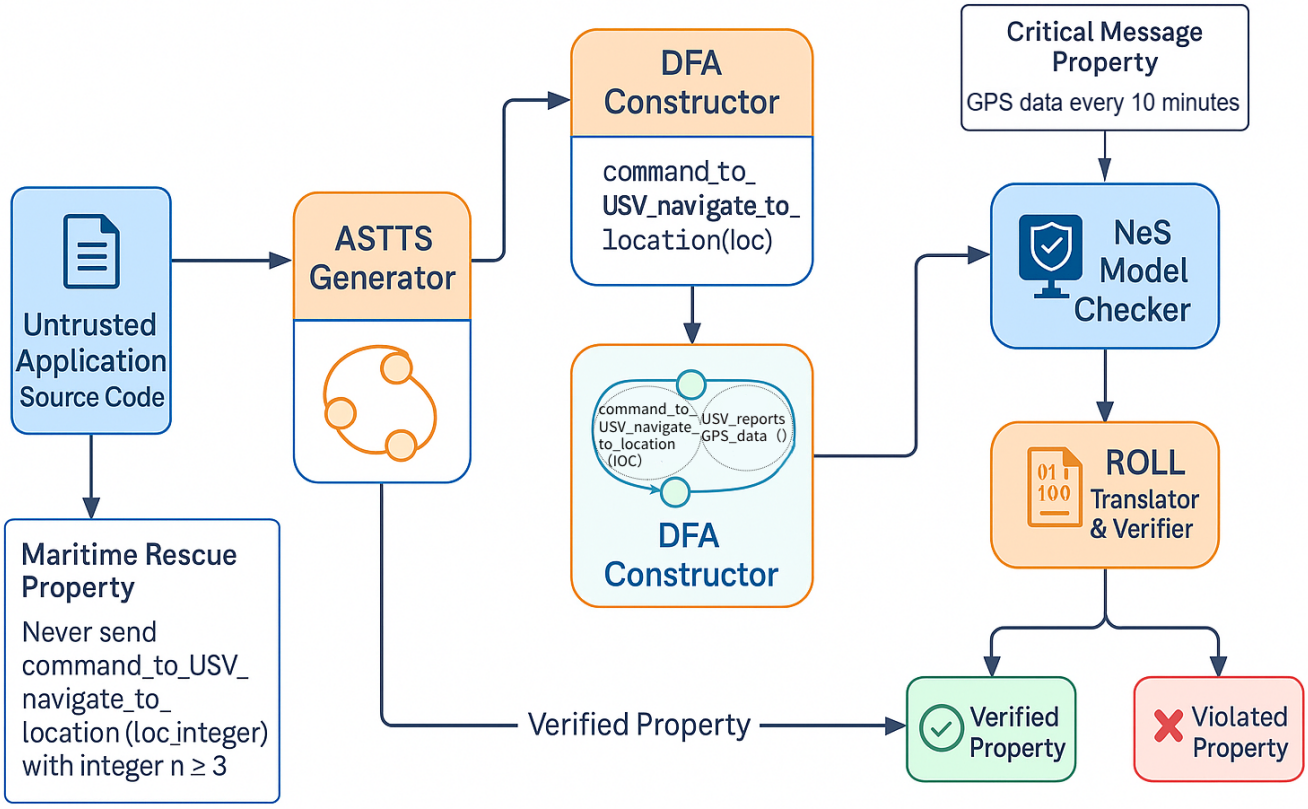

Figure 5. USVScout framework and verification workflow.

`command_to_USV_navigate_to_location(loc:integer)` with integer $n \geq 3$.

Bathymetry Property: The USV must report GPS data at least once every 10 minutes.

USVScout uses $\langle \Phi_0, \Psi_0, \Upsilon_0, \Sigma_0, \Omega_0 \rangle$ to denote the initial DFA that represents the initial state of the whole system. The failure states are represented using all states, i.e., $\mathcal{F}_0 = \Sigma_0$. The initial states are those where the execution starts, i.e., $\mathcal{I}_0 = \Sigma_0$. The transitions between all states are allowed, i.e., $\mathcal{T}_0 = (\Sigma_0, l, \Sigma_0)$. The states in the DFA are all states, i.e., $\Sigma_0 = \Sigma$. The labels of states are empty, i.e., $\Omega_0 = \emptyset$.

A run of the whole system starting from $\langle \Phi_0, \Psi_0, \Upsilon_0, \Sigma_0, \Omega_0 \rangle$ is defined as a function from non-negative integers to states, i.e., $\xi : \mathbb{N} \to \Sigma$. A property specified by $\langle \mathcal{F}, \mathcal{I}, \mathcal{T}, \mathcal{S}, \mathcal{A} \rangle$ is considered violated by run ξ if there exists a run ξ' of $\langle \Phi_0, \Psi_0, \Upsilon_0, \Sigma_0, \Omega_0 \rangle$ such that $\xi' = \xi \circ \xi'$, and there exists $k \in \mathbb{N}$ with $k \geq 0$ such that $\xi'(k) \in \mathcal{F}$. A property specified by $\langle \Phi, \Psi, \Upsilon, \Sigma, \Omega \rangle$ is said to hold on run ξ if it is not violated by ξ. A property specified by $\langle \Phi, \Psi, \Upsilon, \Sigma, \Omega \rangle$ is said to hold on a programming language if it holds on all runs ξ. A system is defined as secure if and only if all relevant security properties hold on it.

The ASTTS created from the source code is used for safety verification. NeS in [10] is used to translate the DFA into a Regular Omega Language Learning (ROLL) script. Then ROLL is used to verify the property. NeS uses the 1- counter LTL formulation. In the 1-counter LTL formulation, properties have the form of $\mathbf{X}\psi_1 \vee (\psi_2 \mathbf{U} \psi_3)$ and $\mathbf{X}(\psi_1 \mathbf{U} \psi_2 \vee \psi_3)$, where ψ_1, ψ_2, and ψ_3 are predicates and $\vee$, $\mathbf{U}$, and $\vee$ are logical operators. Properties are translated into ROLL scripts as shown in [10].

NeS is able to verify properties with the 1-counter LTL formulation. The NeS' verification time increases exponentially as the number of variables in the specification increases.

Most relevant security properties can be translated into properties with the 1-counter LTL formulation. Examples of properties that can be translated into properties with the 1-counter LTL formulation are shown below:

Critical communication.

Let $L > 0$ be the maximum location of a dangerous command. The property is specified as follows:

$$\mathbf{X}\psi_1 \vee (\psi_2 \mathbf{U} \psi_3)$$

where

$$\psi_1 = \langle \psi_{11} \vee \psi_{12} \vee \cdots \vee \psi_{1n} \rangle \wedge \langle \psi_{11} \wedge \psi_{12} \wedge \cdots \wedge \psi_{1n} \rangle,$$
$$\psi_2 = \langle \psi_{21} \vee \cdots \vee \psi_{2L} \rangle,$$
$$\psi_3 = \langle \psi_{2(L+1)} \vee \cdots \vee \psi_{2N} \rangle,$$

and

$$\psi_{1n} = \langle \neg \texttt{command_to_USV_navigate_to_location}(L{+}1) \rangle \wedge \langle \neg \texttt{command_to_USV_navigate_to_location}(L{+}2) \rangle \wedge \cdots,$$
$$\psi_{1k} = \langle \neg \texttt{command_to_USV_navigate_to_location}(k{+}1) \rangle \wedge \langle \neg \texttt{command_to_USV_navigate_to_location}(k{+}2) \rangle \wedge \cdots$$
$$\psi_{2k} = \langle \texttt{command_to_USV_navigate_to_location}(k) \rangle.$$

Critical messages.

Let $T > 0$ be the maximum time interval in which the USV should report collected data. The property is specified as:

$$\mathbf{X}(\psi_1 \mathbf{U} \psi_2 \vee \psi_3)$$

where

$$\psi_1 = \langle \psi_{11} \vee \cdots \vee \psi_{1(T-1)} \rangle,$$
$$\psi_2 = \langle \psi_{1T} \rangle,$$
$$\psi_{1k} = \langle \neg \texttt{USV_reports_GPS_data()} \rangle \wedge \cdots \wedge \langle \neg \texttt{USV_reports_GPS_data()} \rangle.$$

Power management.

The property is specified as:

$$\mathbf{X}(\psi_1 \vee \psi_2 \vee \psi_3 \vee \psi_4 \vee \psi_5)$$

where

$$\psi_1 = \langle \neg \texttt{turn_off_command_to_USV_from_remote_PU} \rangle,$$
$$\psi_2 = \langle \neg \texttt{authenticate_command_to_USV_from_remote_PU} \rangle,$$
$$\psi_3 = \langle \texttt{authenticate_command_to_USV_from_remote_PU} \rangle,$$
$$\psi_4 = \langle \neg \texttt{turn_off_command_to_USV_from_remote_PU} \rangle \wedge \langle \texttt{authenticate_command_to_USV_from_remote_PU} \rangle,$$
$$\psi_5 = \langle \texttt{turn_off_command_to_USV_from_remote_PU} \rangle \wedge \langle \texttt{authenticate_command_to_USV_from_remote_PU} \rangle.$$

Multiple USVs.

The property is specified as:

$$\mathbf{X}(\psi_1 \mathbf{U} \psi_2 \vee \psi_3)$$

where

$$\psi_1 = \langle \psi_{11} \vee \psi_{12} \vee \cdots \vee \psi_{1(N-1)} \rangle,$$
$$\psi_2 = \langle \psi_{1N} \rangle,$$
$$\psi_{1k} = \langle \neg \texttt{command_to_USV_navigate_to_location(loc) with more than 1 USV} \rangle \wedge \langle \neg \texttt{command_to_USV_move_to_location(loc) with more than 1 USV} \rangle \wedge \cdots.$$

The ASTTS also helps to detect a class of vulnerabilities that is called the **programming language bug**.For example, a programmer may accidentally use a loop that has no exit condition, causing the program to enter an infinite loop.The ASTTS is able to detect such bugs because it contains information about the loops in the source code.

6 Evaluation

6.1 Applications

USVScout is used to evaluate the security of three real-world USV-based applications: maritime rescue, waterway maintenance, and bathymetry (see Figure 6). All three applications are implemented in C++ and exhibit a similar architectural design. Each application employs a remote processing unit (PU) to control the USV via a command-line interface (CLI), and utilizes a message queue for status reporting. The bathymetry application adopts Wi-Fi for message transmission, while the other two rely on a simulated message queue. All applications include an onboard PU: in maritime rescue, it is operated by ship crew; in waterway maintenance, it is located on shore; and in bathymetry, it is unused. In addition, each application integrates an aerial vehicle to support its tasks—specifically, an uncrewed aerial vehicle (UAV) is used in maritime rescue and waterway maintenance, while a manned aircraft is employed in bathymetry. Each application runs on a Linux-based operating system (see Table 5) and incorporates one or more sensors and actuators. For further architectural details, refer to section 2.

6.2 Security Properties

The following security properties are relevant to the applications.

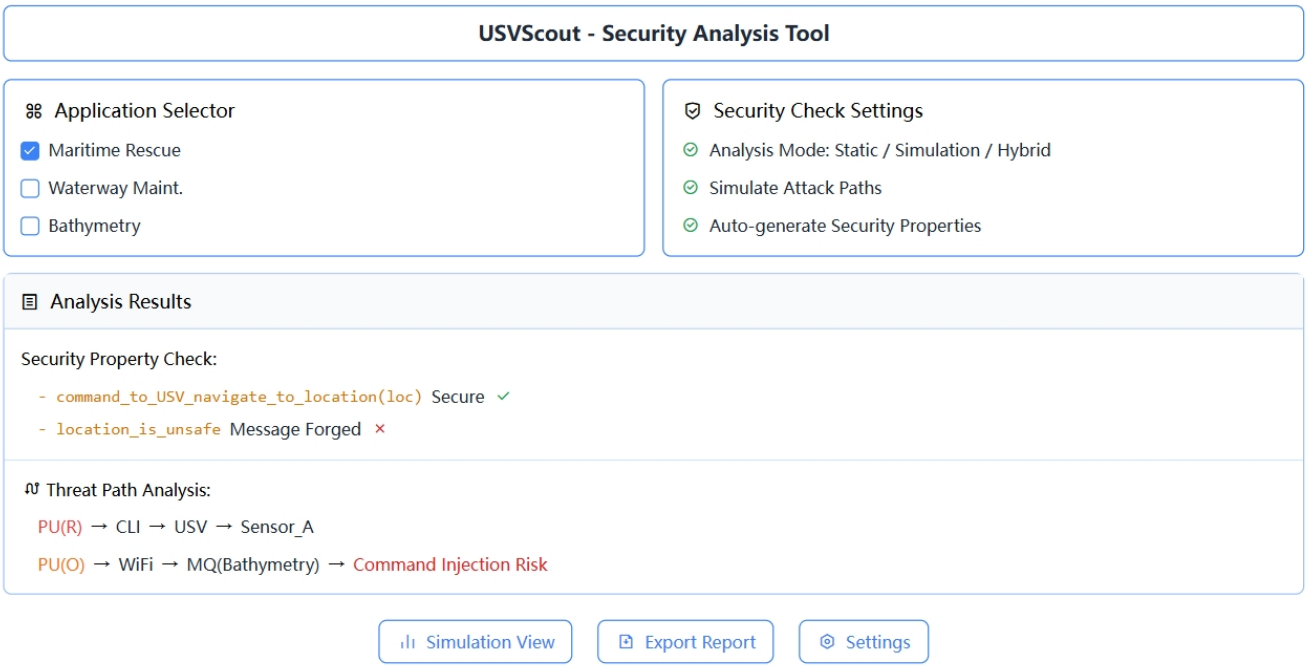

Figure 6. USVScout safety analysis tool interface.

Table 5. Operating systems used in various applications.

Application	OS
Maritime rescue	Linux
Waterway maintenance	Linux
Bathymetry	Linux

6.2.1 *Maritime Rescue*

The critical communications are (1) `command_to_USV_navigate_to_location(loc:integer)`, and (2) `command_to_USV_move_to_location(loc:integer)`. The critical messages are messages with context `location_is_safe` and `location_is_unsafe`. The maximum location of a dangerous command is 3. The maximum time interval in which the USV should report collected data is 10 minutes. The maximum battery level is 10. The battery level decreases by 1 when the remote PU sends a command to the USV and by 1 when the USV reports its status. The USV may stop running when the battery level is low. A location that the USV should not reach is an location with a iceberg. A location is safe if it does not have an iceberg. The remote PU must specify if a command has dangerous semantics when sending `command_to_USV_navigate_to_location(loc:integer)`. The remote PU must specify if a command has dangerous semantics when sending `command_to_USV_move_to_location(loc:integer)`.The remote PU must be authenticated when sending `command_to_USV_navigate_to_location(loc:integer)`, `command_to_USV_move_to_location(loc:integer)`, and `command_to_USV_start_search_for_crew`. The remote PU must report if the UAV has returned when sending `command_to_USV_start_search_for_crew`.
The USV should reach a safe location before reaching the original location. The USV should reach a safe location before running out of battery.The USV should report its position every 10 minutes.

The following three security properties are relevant for maritime rescue:

Property 1. The USV should not navigate to a location with an iceberg.

Property 2. The USV should reach a safe location before reaching a dangerous location.

Property 3. The USV should report its position every 10 minutes.

6.2.2 *Waterway Maintenance*

The critical communication is `command_to_USV_start_dredging`. The maximum time interval in which the USV should reach the safe location is 1 day. The maximum battery level is 10. The battery level decreases by 1 when the remote PU sends a command to the USV and by 1 when the USV reports its status. The USV may stop running when the battery level is low. A location that the USV should not start dredging is a location where the UAV

Table 6. Security properties and verification time for maritime rescue.

Property	Specification	Verification Time (s)
Navigate to a safe location before reaching a dangerous location	$\sim \mathbf{X} \begin{pmatrix} \neg \texttt{command_to_USV_navigate_to_location(3)} \vee \\ \neg \texttt{command_to_USV_navigate_to_location(4)} \vee \cdots \vee \\ \neg \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{USV_reports_GPS_data()} \vee \\ \texttt{USV_reports_GPS_data() with authentication} \end{pmatrix}$	1.31
Reach a safe location before running out of battery	$\sim \mathbf{X} \begin{pmatrix} \neg \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{USV_reports_GPS_data()} \vee \\ \texttt{USV_reports_GPS_data() with authentication} \vee \\ (\neg \texttt{power_level_is_low} \vee \texttt{power_level_is_low}) \vee \\ (\neg \texttt{location_is_safe} \vee \texttt{location_is_safe}) \end{pmatrix}$	1.15
Report GPS data at least every 10 minutes	$\sim \mathbf{X} \left(\psi_1 \mathbf{U} \psi_2 \vee \psi_3\right), \psi_1 = \neg \texttt{USV_reports_GPS_data()} \wedge \cdots \wedge \neg \texttt{USV_reports_GPS_data()}\ \psi_2 = \texttt{USV_reports_GPS_data()}$	1.26

Table 7. Security properties and verification time for waterway maintenance.

Property	Specification	Verification Time (s)
Do not start dredging if UAV has returned	$\sim \mathbf{X} \begin{pmatrix} \neg \texttt{dredging_is_started} \vee \\ \texttt{location_is_safe} \vee \\ \texttt{location_is_unsafe} \vee \\ \texttt{UAV_returns} \vee \\ \texttt{dredging_is_started} \vee \\ \texttt{USV_reports_GPS_data()} \end{pmatrix}$	2.38
Reach a safe location before running out of battery	$\sim \mathbf{X} \begin{pmatrix} \neg \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{USV_reports_GPS_data()} \vee \\ \texttt{USV_reports_GPS_data() with authentication} \end{pmatrix} \vee (\neg \texttt{power_level_is_low} \vee \texttt{power_level_is_low})$	1.14

has not returned. A location is safe if it does not have humans or animals.

The remote PU must authenticate itself when sending `command_to_USV_start_dredging` if the UAV has not returned. The USV should reach a safe location before running out of battery. The remote PU may turn off the USV only when the USV should be turned off. The following two security properties are relevant for waterway maintenance:

Property 1. The USV should not start dredging if the UAV has not returned.

Property 2. The USV should reach a safe location before running out of battery.

6.2.3 Bathymetry

The critical communication is `command_to_USV_collect_data_at_location(loc)`. The critical message is a message that contains semantics `location_is_safe` and `location_is_unsafe`. The maximum battery level is 10. The battery level decreases by 1 when the remote PU sends a command to the USV. The USV may stop running when the battery level is low. A location that the USV should not collect data is an unsafe location.

A location is safe if it does not have boats. The remote PU must authenticate itself when sending `command_to_USV_collect_data_at_location(loc)`. The USV should reach a safe location before running out of battery. The remote PU may turn off the USV only when the USV should be turned off. The USV should report its GPS data at least every 10 minutes. The following two security properties are relevant for bathymetry:

Property 1. The USV should report its GPS data at least every 10 minutes.

Property 2. The remote PU may turn off the USV only when the USV may be turned off.

6.2.4 Power Management

The USV may be remotely shut down by the remote PU or automatically power off due to a low battery level. To ensure safe operation, the USV must only be turned off under appropriate conditions. Specifically, the remote PU is permitted to issue a `turn_off_command` only when the USV is in a state where shutdown is safe and necessary. Moreover, the remote PU must authenticate itself before issuing such a command to prevent unauthorized shutdowns. In scenarios involving battery depletion, the USV must reach a designated safe location prior to shutdown. Based on these requirements, the following security property is defined for power management:

Property 1. The remote PU may turn off the USV only when the USV may be turned off.

6.2.5 Multiple USVs

The remote PU may be capable of controlling multiple USVs simultaneously. In such cases, it may issue commands to more than one USV at a time. However, this control should be exercised only when operationally necessary, as simultaneous commands to multiple USVs may introduce safety risks or indicate abnormal behavior. To mitigate such risks, we define the following security property for coordinated multi-USV control:

Property 1. The remote PU may control multiple USVs only when necessary.

6.3 Experiment

The following USVScout settings are used when checking the above security properties. The order of conjunction and disjunction is Negation Normal Form (NNF). The number of state variables is 11. The number of transition functions is 118. The bound for the state set is 81,944. The bound for the transition functions is 262,144. The laptop has an Intel(R) CPU E3-8250M @ 2.40GHz processor, 8 GB RAM, and Ubuntu 20.04. The version of NeS used is 1.0.0. The OS is Ubuntu 20.04. The Java version is 1.8. The memory used is determined by the specification of the property. All experiments were run on the laptop.

USVScout is applied to check the above security properties. For each application, the most recent version available on the web is used, and the code base is not modified. Table 6 shows the specifications of relevant security properties and verification time for the maritime rescue system. Table 7 shows the specifications of relevant security properties and verification time for the waterway maintenance system. Table 8 shows the specifications of relevant security properties and verification time for the bathymetry system. Table 9 shows the specifications of relevant security properties and verification time for the power management component. Table 10 shows the specifications of relevant security properties and verification time for the multiple usv scenario. All security properties are satisfied by version 1.0.0 of the source code.

7 Limitation

USVScout is an analysis tool specifically designed for applications based on unmanned surface vehicles (USVs), and thus its application scope has certain limitations. Firstly, this tool is not applicable to systems that do not involve USV, such as ship-to-shore (V2S) communication applications without USV participation. Even in USV-based systems, USVScout is only suitable for application scenarios where the USV is remotely and actively controlled, and does not support deployment methods with passive monitoring as the main function, such as Marine environment surveillance and other application scenarios.In terms of threat modeling, USVScout mainly analyzes those explicit attacker roles that are explicitly reflected in the application logic or can be modeled through the framework. This tool is unable to handle security issues involving covert attackers, such as attack behaviors carried out by exploiting unknown operating system-level vulnerabilities.In addition, USVScout also has certain limitations on the types of security attributes that can be specified and verified. It currently only supports security attributes that can be encoded through single-counter linear temporal logic (1-counter LTL) formulas, but does not support attributes that require more complex

Table 8. Security properties and verification time for bathymetry.

Property	Specification	Verification Time (s)
Report GPS data at least every 10 minutes	$\sim \mathbf{X}\,(\psi_1\,\mathbf{U}\,\psi_2 \vee \psi_3)$, where: $\psi_1 = \neg$ `USV_reports_GPS_data()` $\wedge \cdots \wedge \neg$ `USV_reports_GPS_data()` $\psi_2 =$ `USV_reports_GPS_data()`	1.21
Remote PU may turn off USV only when USV is at a safe location	$\sim \mathbf{X} \begin{pmatrix} \neg \texttt{turn_off_command_to_USV_from_remote_PU} \vee \\ (\texttt{location_is_safe} \vee \texttt{location_is_unsafe}) \vee \\ \texttt{turn_off_command_to_USV_from_remote_PU} \end{pmatrix}$	1.38

Table 9. Security properties and verification time for power management.

Property	Specification	Verification Time (s)
Remote PU may turn off USV only when USV may be turned off	$\sim \mathbf{X} \begin{pmatrix} \neg \texttt{turn_off_command_to_USV_from_remote_PU} \vee \\ (\texttt{location_is_safe} \vee \texttt{location_is_unsafe}) \vee \\ \texttt{turn_off_command_to_USV_from_remote_PU} \end{pmatrix}$	1.38

Table 10. Security properties and verification time for multiple USVs.

Property	Specification	Verification Time (s)
Remote PU may control multiple USVs only when necessary	$\sim \mathbf{X} \begin{pmatrix} \neg \texttt{command_to_USV_navigate_to_location(loc) with more than 1 USV} \vee \\ \neg \texttt{command_to_USV_move_to_location(loc) with more than 1 USV} \vee \\ \neg \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{authenticate_command_to_USV_from_remote_PU} \vee \\ \texttt{USV_reports_GPS_data()} \end{pmatrix}$	1.22

reasoning mechanisms, such as properties involving global induction or circular dependencies.Finally, from the implementation perspective, USVScout currently only supports programs written in C++ because it relies on the intermediate representation verification toolchain specifically designed for C++ source code. This tool does not yet provide support for applications developed in other programming languages.

USVScout may not be able to identify certain vulnerabilities that can be exploited under the current threat model. This limitation mainly occurs because the exploitability of vulnerabilities is closely related to the specific implementation of the application and the threat assumptions defined in the model. For example, consider the vulnerabilities caused by the use of loop construction in the absence of appropriate exit conditions. If the application does indeed use such a structure and the threat model contains attackers who can trigger the loop, then the vulnerability may be exploitable. However, due to the abstraction level of USVScout, it may not be able to detect this issue. On the contrary, if the application does not use such a structure, there is no vulnerability, and USVScout can correctly conclude that it does not exist. Similarly, vulnerabilities related to unverified user input may or may not be considered exploitable, depending on the defined threat capability. If the threat model assumes that attackers can provide arbitrary input, then this vulnerability may be exploitable. On the contrary, if these attack vectors are outside the model's scope, the vulnerability cannot be triggered and the assessment of

USVScout remains valid. These examples emphasize that the effectiveness of USVScout essentially depends on the interaction between the application control flow and the accuracy of the threat modeling process.

8 Conclusion

A novel threat model for Unmanned Surface Vehicle-based (USV) applications is proposed, which is capable of depicting adversarial behaviors beyond the scope of traditional security assumptions. Based on this model, USVScout—an analysis framework for automatically verifying security attributes in USV systems—was designed and implemented. USVScout was applied to three real-world maritime systems, revealing 12 previously undisclosed security vulnerabilities. These vulnerabilities are exploitable under the proposed threat model, but none have been identified by existing static analysis tools. Experiments show that USVScout can complete the detection of violations within a few seconds on an ordinary laptop, demonstrating high efficiency and practicality. In conclusion, USVScout has proven to be effective for conducting security analysis on USV-based applications in real-world threat scenarios and can identify potential security risks along critical paths.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

Shihao Zhou is an employee of Three Gorges Navigation Authority, Yichang, China.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Gallego, A. J., Pertusa, A., Gil, P., & Fisher, R. B. (2019). Detection of bodies in maritime rescue operations using unmanned aerial vehicles with multispectral cameras. *Journal of Field Robotics, 36*(4), 782-796. [Crossref]

[2] Specht, M. (2024). Methodology for performing bathymetric and photogrammetric measurements using UAV and USV vehicles in the coastal zone. *Remote Sensing*, *16*(17), 3328. [Crossref]

[3] Jung, S., Cho, H., Kim, D., Kim, K., Han, J. I., & Myung, H. (2017). Development of algal bloom removal system using unmanned aerial vehicle and surface vehicle. *IEEE Access*, *5*, 22166-22176. [Crossref]

[4] Li, W., Ge, Y., Guan, Z., & Ye, G. (2022). Synchronized motion-Based UAV–USV cooperative autonomous landing. *Journal of Marine Science and Engineering, 10*(9), 1214. [Crossref]

[5] de Carnavalet, X. D. C., & Mannan, M. (2016, February). Killed by proxy: Analyzing client-end TLS interception software. In *Network and Distributed System Security Symposium* (pp. 21-24). [Crossref]

[6] Zainudin, A., Putra, M. A. P., Alief, R. N., Kim, D. S., & Lee, J. M. (2024, June). Blockchain-aided collaborative threat detection for securing digital twin-based IIoT networks. In *ICC 2024-IEEE International Conference on Communications* (pp. 4656-4661). IEEE. [Crossref]

[7] Jia, Y., Yuan, B., Xing, L., Zhao, D., Zhang, Y., Wang, X., ... & Jin, H. (2021, November). Who's in control? on security risks of disjointed IoT device management channels. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1289-1305). [Crossref]

[8] Wang, Q., Ji, S., Tian, Y., Zhang, X., Zhao, B., Kan, Y., ... & Beyah, R. (2021). MPInspector: A systematic and automatic approach for evaluating the security of IoT messaging protocols. In *30th USENIX Security Symposium* (*USENIX Security 21*) (pp. 4205-4222).

[9] Wang, M., Tian, C., Zhang, N., Duan, Z., & Yao, C. (2020). Translating Xd-C programs to MSVL programs. *Theoretical Computer Science, 809*, 430-465. [Crossref]

[10] Li, Y., & Li, L. (2012). Model checking of linear-time properties based on possibility measure. *IEEE Transactions on Fuzzy systems, 21*(5), 842-854. [Crossref]

[11] Celik, Z. B., McDaniel, P., & Tan, G. (2018). Soteria: Automated IoT safety and security analysis. In *2018 USENIX annual technical conference* (*USENIX ATC 18*) (pp. 147-158).

[12] Wang, Q., Datta, P., Yang, W., Liu, S., Bates, A., & Gunter, C. A. (2019, November). Charting the attack surface of trigger-action IoT platforms. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security* (pp. 1439-1453). [Crossref]

[13] Celik, Z. B., Tan, G., & McDaniel, P. (2019). IOTGUARD: Dynamic Enforcement of Security and Safety Policy in Commodity IoT. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019*. The Internet Society. [Crossref]

[14] Yuan, B., Wu, Y., Yang, M., Xing, L., Wang, X., Zou, D., & Jin, H. (2022). Smartpatch: Verifying the authenticity of the trigger-event in the iot platform. *IEEE Transactions on Dependable and Secure Computing, 20*(2), 1656-1674. [Crossref]

[15] Tam, K., & Jones, K. (2018, June). Cyber-risk assessment for autonomous ships. In *2018 international*

conference on cyber security and protection of digital services (cyber security) (pp. 1-8). IEEE. [Crossref]

[16] Akpan, F., Bendiab, G., Shiaeles, S., Karamperidis, S., & Michaloliakos, M. (2022). Cybersecurity challenges in the maritime sector. *Network, 2*(1), 123-138. [Crossref]

[17] Caprolu, M., Di Pietro, R., Raponi, S., Sciancalepore, S., & Tedeschi, P. (2020). Vessels cybersecurity: Issues, challenges, and the road ahead. *IEEE Communications Magazine, 58*(6), 90-96. [Crossref]

[18] Radoš, K., Brkić, M., & Begušić, D. (2024). Recent advances on jamming and spoofing detection in GNSS. *Sensors, 24*(13), 4210. [Crossref]

[19] Aslam, S., Michaelides, M. P., & Herodotou, H. (2020). Internet of ships: A survey on architectures, emerging applications, and challenges. *IEEE Internet of Things journal, 7*(10), 9714-9727. [Crossref]

[20] Rath, S., Intriago, A., Sengupta, S., & Konstantinou, C. (2023, August). Lost at sea: Assessment and evaluation of rootkit attacks on shipboard microgrids. In *2023 IEEE Electric Ship Technologies Symposium (ESTS)* (pp. 534-541). IEEE. [Crossref]

[21] Tullsen, M., Pike, L., Collins, N., & Tomb, A. (2018, July). Formal verification of a vehicle-to-vehicle (V2V) messaging system. In *International Conference on Computer Aided Verification* (pp. 413-429). Cham: Springer International Publishing. [Crossref]

[22] Grigoriadis, C., Papastergiou, S., Kotzanikolaou, P., Douligeris, C., Dionysiou, A., Elias, A., ... & Kamm, L. (2021, August). Integrating and validating maritime transport security services: Initial results from the cs4eu demonstrator. In *Proceedings of the 2021 Thirteenth International Conference on Contemporary Computing* (pp. 371-377). [Crossref]

[23] Zhang, C., Cao, C., Kang, K., Guo, C., & Guo, M. (2022). Virtual global positioning system construction approach for unmanned surface vessel based on Dempster–Shafer theory and broad learning framework. *The Journal of Navigation, 75*(5), 1144-1166. [Crossref]

[24] Sun, X., Wang, G., Fan, Y., Mu, D., & Qiu, B. (2018). An automatic navigation system for unmanned surface vehicles in realistic sea environments. *Applied Sciences, 8*(2), 193. [Crossref]

[25] Akram, W., Yang, S., Kuang, H., He, X., Din, M. U., Dong, Y., ... & Hussain, I. (2024). Long-Range Vision-Based UAV-assisted Localization for Unmanned Surface Vehicles. *arXiv preprint arXiv:2408.11429*.

[26] Boretti, A. (2024). Unmanned surface vehicles for naval warfare and maritime security. *The Journal of Defense Modeling and Simulation*, 15485129241283056. [Crossref]

[27] Berbecaru, D. G., & Lioy, A. (2021, September). Attack strategies and countermeasures in transport-based time synchronization solutions. In *International Symposium on Intelligent and Distributed Computing* (pp. 203-213). Cham: Springer International Publishing. [Crossref]

[28] Hashali, S. D., Yang, S., & Xiang, X. (2024). Route planning algorithms for unmanned surface vehicles (USVs): a comprehensive analysis. *Journal of Marine Science and Engineering, 12*(3), 382. [Crossref]

[29] He, P., Du, X., Li, Y., Guo, H., & Cui, J. (2025). An integration methodology of safety and security requirements for autonomous vehicles. *Journal of Transportation Safety & Security, 17*(3), 253-271. [Crossref]

[30] Hofer-Schmitz, K., & Stojanović, B. (2020). Towards formal verification of IoT protocols: A Review. *Computer Networks, 174*, 107233. [Crossref]

[31] Cai, X., Shi, K., She, K., Zhong, S., Wen, S., & Xie, Y. (2023). Communication security of autonomous ground vehicles based on networked control systems: The optimized LMI approach. *Security and Safety, 2*, 2023016. [Crossref]

[32] Wang, H., Ren, G., Chen, J., Ding, G., & Yang, Y. (2018). Unmanned aerial vehicle-aided communications: Joint transmit power and trajectory optimization. *IEEE Wireless Communications Letters, 7*(4), 522-525. [Crossref]

[33] Jero, S., Hoque, E., Choffnes, D., Mislove, A., & Nita-Rotaru, C. (2018, July). Automated Attack Discovery in TCP Congestion Control Using a Model-guided Approach. In *Proceedings of the 2018 Applied Networking Research Workshop* (pp. 95-95). [Crossref]



Shihao Zhou received the B.S. degree in Computer Science and Technology from Wuhan University of Science and Technology, China, in 2017. (Email: 2772613724@qq.com)