drones

MDPI

Article

Searching for Scalable Networks in Unmanned Aerial Vehicle Infrastructure Using Spatio-Attack Course-of-Action

Seok Bin Son [1,*] and Dong Hwa Kim [2]

[1] School of Electrical Engineering, Korea University, Seoul 02841, Republic of Korea
[2] Agency for Defense Development (ADD), Seoul 05661, Republic of Korea; dhkim@add.re.kr
* Correspondence: lydiasb@korea.ac.kr

Abstract: Unmanned aerial vehicles are increasingly being applied to various applications for a variety purposes, such as delivery, communication relay, mapping, and surveillance services. Through these, it is possible to provide flexible and stable network services. Unmanned aerial vehicles perform a wide range of tasks using Internet-of-Things technology, which needs Internet access. These internet connections, however, make it more possible for attackers to execute various security attacks on unmanned aerial vehicles. Therefore, it is crucial to identify the attack behavior of the adversary, which is called "course-of-action", to preserve security in the unmanned aerial vehicle infrastructure. Based on learned data, the existing course-of-action method has the drawback of not functioning on various networks. As a result, in this paper, we propose a novel heuristic search-based algorithm to apply to various unmanned aerial vehicle infrastructures. The algorithm can build the optimal heuristic functions in various unmanned aerial vehicle network environments to explore the attack course-of-action and design the optimal attack paths to maximize total reward. Applying the proposed algorithm in two unmanned aerial vehicle network scenarios allowed us to confirm that the best attack path is well established.

Keywords: course-of-action; course-of-action attack; cyber security; drones; heuristic function; scalable network; spatio-attack course-of-action algorithm; unmanned aerial vehicles

Citation: Son, S.B.; Kim, D.H. Searching for Scalable Networks in Unmanned Aerial Vehicle Infrastructure Using Spatio-Attack Course-of-Action. *Drones* **2023**, *7*, 249. https://doi.org/10.3390/drones7040249

Academic Editor: Higinio González Jorge

Received: 25 February 2023
Revised: 15 March 2023
Accepted: 27 March 2023
Published: 4 April 2023

1. Introduction

Nowadays, it is feasible to operate and configure various networks by constructing mobile infrastructure utilizing unmanned aerial vehicles (UAVs) [1]. UAVs or drones, as their name implies, are aircraft that fly autonomously under the control of computer software or via remote control [2]. UAVs are being employed for several purposes, including delivery, communication relay, and mapping, and this an increasing need for them [3,4]. In particular, using UAVs can provide reliable surveillance services and provide flexible and reliable network services, and various studies using them are actively being conducted [5–9]. Since all UAV components are connected, IoT technology is required to accomplish numerous functionalities using these UAVs [10]. However, due to this internet connection, various security threats to UAVs can occur if such UAV communication is hacked and used by the attacker. In particular, as shown in Figure 1, security threats, such as deauthentication attack, denial-of-service attack (DoS attack), and snooping attack, can occur in UAVs. Therefore, security is becoming increasingly crucial in UAV situations as the network grows more complicated and extended [11,12].

A cyber security technique examines the network environment to evaluate system security or identify vulnerabilities, which is often called "penetration testing" or "course-of-action (COA)". To comprehend the behavior of potential attackers in these UAV infrastructures, it is necessary to approach the attackers' activity. As a result, in this paper, we used attack COA to verify the security of the UAV infrastructure and to prepare for attack behavior. To design the attack COA search algorithm, various state-of-the-art artificial intelligence (AI)

methods can be used, such as deep and distributed learning [13–16], reinforcement learning (RL) [7,17,18], etc. Most recent research has employed deep learning-based approaches.

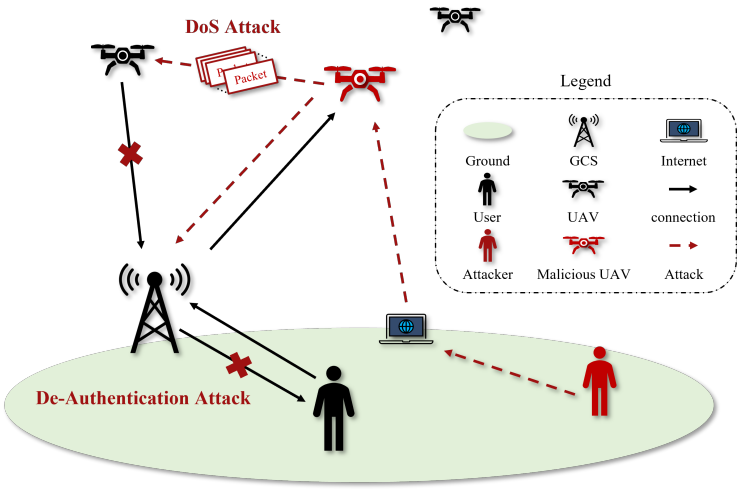


Figure 1. An example of the attacks on UAV infrastructure.

However, they have problems in that they do not account for the conditions of various networks since they operate on only learned data. Therefore, these existing algorithms cannot be scalable. In this paper, to overcome these limitations, we propose a new algorithm, the "spatio-attack COA algorithm", based on the heuristic search, among various heuristic methods [19]. Our suggested algorithm can search for an attack COA and plan paths in various types of networks by constructing optimum heuristic functions. Our algorithm's structure is rooted in the A-star search algorithm, which focuses on finding the shortest path between the source and target nodes in scalable networks. However, unlike the traditional A-star search algorithm, our algorithm is designed for maximizing total reward as a Markov decision process (MDP). The major contribution of this paper are as follows:

- The proposed algorithm operates automatically with no data issues in scalable networks.
- The proposed algorithm can maximize total rewards while finding the optimal attack path.
- The proposed algorithm was applied to two UAV network scenarios to verify that the optimal attack path is well established.

The rest of this paper is constructed as follows. Section 2 describes the security threats in UAV infrastructures, and Section 3 identifies our proposed algorithm, the "spatio-attack COA search". In addition, Section 4 displays the performance evaluation. Finally, Section 5 presents this paper's conclusion and outlines future work.

2. Preliminaries

The associated models, methods, and prior research results related to the proposed algorithm are described in this section. In Section 2.1, we discuss the security threat models in UAV infrastructure, and in Section 2.2, we present the existing COA attack search methods. Finally, the MDP-based algorithm is described in Section 2.3.

2.1. Security Threat Models in UAV Infrastructure

As communication using UAVs has become more complex and diverse, the possibility of security threats has increased. The attacker may exploit the code of the software to hack sensitive data on the user's device, which can cause serious adverse effects on communication in applications using UAVs [2]. Examples of representative security threats that can be posed to UAVs are as follows: deauthentication attack, DoS attack, and snooping attack. The details of each attack are described below.

Most UAV communications are based on the 802.11 protocol, Wi-Fi, which allows communication between the user and the ground control station (GCS). However, since the Wi-Fi is not encrypted with the management authentication system, the attacker may be able to access and take advantage of it without much difficulty [20]. In this case, the attacker may transmit the manipulated authentication frame to the GCS to obstruct the operating UAV connection [21]. The attacker may now control the UAV through this.

In addition, UAVs are especially vulnerable to DoS attacks since UAVs use limited computing resources and power. A DoS attack is the traditional flooding attack [22], in which the user's network system is used to consume and deplete resources for the attacker's malicious goals, prohibiting the user from using those resources for intended and regular applications [20]. Existing studies have been done to carry out DoS attacks on UAVs [20,22], and if many attack packets are sent to UAVs, the user's system becomes paralyzed, and resources are insufficient. UAVs may not receive commands from the GCS, and an accident may occur [20].

The snooping attack occurs when a malicious attacker pretends to be a valid user and secretly bypasses the victim's system to spy on sensitive information unlawfully. Furthermore, snooping attacks may also be executed on UAVs. Since a UAV uses the installed software of a given company, it could be abused for this attack [2]. For example, SensePost announced that the attacker could hack UAVs for malicious purposes and snoop data from the victim's smartphone [2]. It has been mentioned that this attack can be particularly effective in crowded places with a density of smartphone users [23,24].

2.2. Existing COA Attack Search Methods

COA attacks are used in cyber attack simulation technologies to discover prospective attack risk factors by evaluating the organization's policies and status and implementing security improvement plans. As a result, COA attack has become a crucial tool in cyber security for performing cyber analytics and exploring and analyzing a collection of security issues [25].

On the other hand, existing approaches of manually configuring cyber security threat COA attack strategies have several limitations, as shown in Table 1. Traditional COA attack approaches need the cooperation of security specialists with cyber security knowledge, and the effectiveness of their implementation is dependent on their skill. Furthermore, it is time-consuming and costly because cyber security must be examined directly. As a result, technology that can automatically conduct COA attacks is required to improve the efficiency of cyber security research and generate effective defense plans. As a result, various COA attack search methodologies have emerged, including sophisticated COA search and modeling techniques such as attack tree [26], attack graph [27], and game theory [28]. Furthermore, multiple cases of COA search using various AI-based algorithms have recently emerged.

Table 1. Technical comparison between previous COA methods and our spatio-attack COA search.

Method	Automatic	No Data Issues	Maximize Reward	Scalable
Traditional COA attack	X	X	X	X
Attack Tree	O	X	X	X
Attack Graph	O	X	X	X
Reinforcement Learning	O	O	O	X
A-star	O	O	X	O
Spatio-Attack COA	O	O	O	O

Attack graphs and trees are explainable models for evaluating system security and identifying probable attack methods. However, because both methods need comprehensive knowledge of the target network's structure and the configuration of all hosts, they are impossible to use from the standpoint of an actual attacker. Furthermore, due to the increase in state space and complex modeling processes, they are difficult to use in large-

scale network scenarios [29]. The scalability issue arises in the attack graph and attack tree due to the exponential growth in the number of nodes and edges, which becomes a significant problem as the network size increases [30]. This issue is primarily caused by the necessity to consider all the complete attack paths in both attack graphs and attack trees. Additionally, difficulties occur when penetration testing is conducted using game theory. When employing game theory, the risk of not being able to see the setup of all networks and the possibility of deploying unreliable attack tools should be considered. When using game theory for penetration testing, a wholly controlled stochastic game is required, which makes satisfying a real-world network challenging. environment [31]. Furthermore, there is a limit to applying existing AI algorithms in the cyber security field. Variability issues might develop due to a lack of data, and there is a possibility of misleading conclusions being drawn due to data overfitting issues. To overcome these limitations, RL algorithms [32] have appeared in the security field.

2.3. Markov Decision Process (MDP)-Based Algorithm

RL is a machine learning area corresponding to the MDP. As shown in Figure 2, MDP consists of an agent, state, action, policy, environment, and reward. RL is a process where a decision maker observes state information from the environment in which the decision maker exists and probabilistically determines behavior through policies; when such behavior is applied to the environment, the decision maker is rewarded and receives the following state information. In other words, decision-making is achieved by continuously repeating state observation, behavioral decision, state transition, the next state, and the immediate reward. RL makes decisions to maximize the total amount of cumulative rewards available until the system ends. In our environment, we use MDP to make decisions to maximize the cumulative attack value when the agent attacks. In network topology, each node is in a state visited by an agent, and selecting the next node is an action. The agent makes a continuous decision with the reward obtained by evaluating the current state and behavior. To maximize the rewards that an agent receives, finding the optimal policy in each state is necessary. However, as shown in Table 1, since RL algorithms are based on training, there is a problem in that they cannot have scalability.

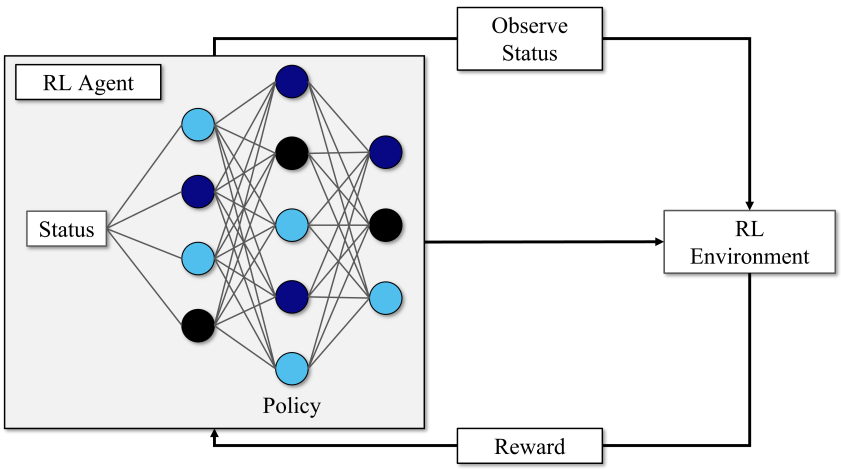


Figure 2. Overview of Markov Decision Process.

3. Spatio-Attack COA Search

To overcome the limits of the previous COA methods, as shown in Table 1, we designed a new method. We have coined the term and algorithm "spatio-attack COA algorithm" to test the UAV infrastructure for various security exploits. The term "spatio" refers to spatial intelligent attack COA heuristic for scalable networks. Our suggested spatio-attack COA algorithm may be used for various types of networks by developing appropriate heuristic functions for attack COA search and attack path planning.

As our aim is to handle scalable networks, our proposed algorithm is especially designed and implemented by utilizing intelligent and adaptive search-based techniques with an appropriate heuristic function definition. In addition, the algorithm's structure is based on the A-star search algorithm [33], which is good at scalable networks. The A-star search algorithm is a popular graphic search algorithm, which is regarded as one of the most significant intelligent search algorithms since it combines breadth-first and depth-first search advantages. It is also a highly efficient heuristic method to locate a variable or low-cost path. It is worth noting that while our algorithm's base structure is the A-star search algorithm, our algorithm is designed to maximize the total reward, while the A-star algorithm only considers the shortest path from the source node to the target node. Therefore, in the spatio-attack COA algorithm, the optimal attack path search process in a network environment can be modeled as the MDP.

As a result, as shown in Table 1, our spatio-attack COA algorithm operates automatically with no data issues in scalable networks and maximizes reward. In other words, our approach incorporates both reward maximization, a feature of reinforcement learning algorithms, and scalability, a feature of A-star algorithms. The detailed descriptions of each part are described below.

We designed the spatio-attack COA algorithm for scalable networks to establish an optimal attack path from the source node, where the attacker is located, to the target node. We first proposed that there is an intermediate node between the source node and the target node. The attack path from the source node to the target node is therefore made up of the attack paths from the source node to the intermediate node and the intermediate node to the target node combined. The value of the attack is calculated as follows for this purpose:

$$f(n) = g(n) + h(n) \tag{1}$$

$$g(n) = \begin{cases} 0 & \text{source node} \\ \sum_{t'=1}^{t} r(s'_t, a'_t) & \text{other nodes} \end{cases} \tag{2}$$

$$h(n) = \sum_{t'=t}^{T} r(s'_t, a'_t) \tag{3}$$

As shown in the Formula (1), we defined the heuristic function $f(n)$ as the attack value at the current middle network node n. In addition, $g(n)$ denotes the addition of attack values from the attack source node to the existing middle network node n. Meanwhile, $h(n)$ is defined as the approximation of the attack values from the current middle network node n to our target node. The components of the formula in detail are described below.

As shown in the Formula (2), $g(n)$ is a value obtained by calculating the sum of edges passed to reach the intermediate node, which is the current node. The attack value in the case of the source node is 0, and the other nodes are $\sum_{t'=1}^{t} r(s'_t, a'_t)$. t denotes a time step as long as it advances to the next node. s and a indicate a state and an action, respectively. $r(s'_t, a'_t)$ means the weight value of the edges passing by when moving from the current node s to the next node s'.

$h(n)$, as shown in the Formula (3), is defined as an approximate value for the attack path from the current node to the target node, which may be considered a heuristic function. In this case, it is essential to accurately design $h(n)$ to identify the optimal attack path. We use vulnerability information and a standard vulnerability scoring system (CVSS) as $h(n)$ for all nodes [34]. CVSS prioritizes responses by scoring vulnerabilities in a standardized way that is mainly used in the security field to determine the severity of vulnerabilities. As shown in Formula (4), CVSS consists of two components: base score and exploitability score. The base Score shows the vulnerability's risk, while the exploitability score indicates the probability of exploiting a specific vulnerability, with the value range reaching up to 10. In this paper, we weigh the base score, considering how feasible it is to use a specific

vulnerability with the explorability score [35]. The detail of the $h(n)$ function's design is as follows:

$$h(n) = Score_{vul} = Base\ Score \times \frac{Exploitability\ Score}{10} \quad (4)$$

We used the corresponding Formula (4) in $h(n)$ as follows. Base score denotes a score between 0 and 10 based on the framework's evaluation of the vulnerability's attributes and severity. The exploitability score assesses the current vulnerability attack technology or code availability state. The public availability of simple exploit codes increases the severity of the vulnerability by allowing inexperienced attackers to exploit it, thereby increasing the number of possible attackers. Assuming that a significant vulnerability would likely reach the target node, we assigned a higher $h(n)$ value to nodes with higher vulnerabilities. In other words, 0.01 and 100 were given for the source and target nodes, respectively. Furthermore, $h(n)$ was assigned using CVSS information to nodes that attack vulnerabilities, 1.5 to nodes that access or execute files using execCode or accessFile, respectively, and 0 to all other nodes.

In the case of $g(n)$, the sum of the weight values of the edges between the nodes was passed by, and the weight value was also constructed using CVSS information of two adjacent nodes, while if it was not connected, –1 was given. In summary, *f(n)* is modeled using vulnerability information and learns to select nodes where attackers receive higher rewards. Therefore, a spatially optimal attack path search is preferentially performed using the corresponding algorithm. As a result, the values of each node in the *f(n)* function are shown in Tables 2 and 3, and the values of the edges are equal to the sum of the values of the two connected nodes.

Table 2. Information of vulnerability in scenario 1.

Node	Description	Value
0	execCode(UAV, root)	100
1	RULE 4 (Trojan horse installation)	0.1
2	accessFile(UAV, '/usr/local/share')	0.1
3	RULE 16 (NFS semantics)	0.1
4	accessFile(fileServer, '/export')	0.1
5	RULE 10 (execCode implies file access)	0.3
6	canAccessFile(fileServer, root, '/export')	0.1
7	execCode(fileServer, root)	0.3
8	RULE 2 (remote exploit of a server program)	1.794
9	netAccess(fileServer, tcp, 80)	0.1
10	RULE 5 (multi-hop access)	0.1
11	hacl(webServer, fileServer, tcp, 80)	0.1
12	execCode(webServer, 'Apache httpd')	0.3
13	RULE 2 (remote exploit of a server program)	3.698
14	netAccess(webServer, tcp, 443)	0.1
15	RULE 6 (direct network access)	0.1
16	hacl(internet, webServer, tcp, 443)	0.1
17	attackerLocated(internet)	0.01
18	networkServiceInfo(webServer, https, tcp, 443, 'Apache httpd')	0.1
19	vulExists(webServer, 'CVE-2015-3185', https, remoteExploit, privEscalation)	3.698
20	networkServiceInfo(fileServer, http, tcp, 80, root)	0.1
21	vulExists(fileServer, 'CVE-2006-3011', http, remoteExploit, privEscalation)	1.794
22	RULE 17 (NFS shell)	0.1
23	hacl(webServer, fileServer, nfsProtocol, nfsPort)	0.1
24	nfsExportInfo(fileServer, '/export', webServer)	0.1
25	nfsMounted(UAV, '/usr/local/share', fileServer, '/export', read)	0.1

Table 3. Information of vulnerability in scenario 2.

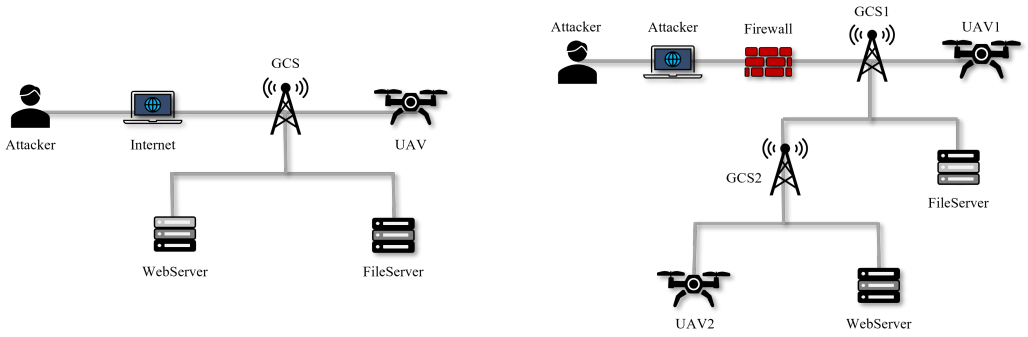
Node	Description	Value	Node	Description	Value
0	execCode(UAV2, root)	100	24	vulExists(fireWall, 'CVE-2012-0883', https, remoteExploit, privEscalation)	2.346
1	RULE 4 (Trojan horse installation)	0.1	25	networkServiceInfo(webServer, https, https, 80, 'Apache httpd')	0.1
2	accessFile(UAV2, '/usr/local/share')	0.1	26	vulExists(webServer, 'CVE-2012-0053', https, remoteExploit, privEscalation)	3.698
3	RULE 16 (NFS semantics)	0.1	27	RULE 5 (multi-hop access)	0.1
4	accessFile(fileServer, '/export')	0.1	28	execCode(webServer, root)	0.3
5	RULE 10 (execCode implies file access)	0.3	29	RULE 4 (Trojan horse installation)	0.1
6	canAccessFile(fileServer, root, '/export')	0.1	30	accessFile(webServer, '/export')	0.1
7	execCode(fileServer, root)	0.3	31	RULE 17 (NFS shell)	0.1
8	RULE 2 (remote exploit of a server program)	1.794	32	hacl(fireWall, webServer, nfsProtocol, nfsPort)	0.1
9	netAccess(fileServer, tcp, 80)	0.1	33	nfsExportInfo(webServer, '/export', fireWall)	0.1
10	RULE 5 (multi-hop access)	0.1	34	networkServiceInfo(fileServer, http, tcp, 80, root)	0.1
11	hacl(webServer, fileServer, tcp, 80)	0.1	35	vulExists(fileServer, 'CVE-2006-3011', http, remoteExploit, privEscalation)	1.794
12	execCode(webServer, 'Apache httpd')	0.3	36	RULE 17 (NFS shell)	0.1
13	RULE 2 (remote exploit of a server program)	3.698	37	hacl(webServer, fileServer, nfsProtocol, nfsPort)	0.1
14	netAccess(webServer, https, 80)	0.1	38	nfsExportInfo(fileServer, '/export', webServer)	0.1
15	RULE 5 (multi-hop access)	0.1	39	RULE 17 (NFS shell)	0.1
16	hacl(fireWall, webServer, https, 80)	0.1	40	nfsMounted(UAV2, '/usr/local/share', fileServer, '/export', read)	0.1
17	execCode(fireWall, 'Apache')	0.3	41	RULE 16 (NFS semantics)	0.1
18	RULE 2 (remote exploit of a server program)	2.346	42	accessFile(mailServer, '/export')	0.1
19	netAccess(fireWall, tcp, 443)	0.1	43	RULE 17 (NFS shell)	0.1
20	RULE 6 (direct network access)	0.1	44	hacl(webServer, mailServer, nfsProtocol, nfsPort)	0.1
21	hacl(internet, fireWall, tcp, 443)	0.1	45	nfsExportInfo(mailServer, '/export', webServer)	0.1
22	attackerLocated(internet)	0.01	46	RULE 17 (NFS shell)	0.1
23	networkServiceInfo(fireWall, https, tcp, 443, 'Apache')	0.1	47	nfsMounted(UAV2, '/usr/local/share', mailServer, '/export', read)	0.1

4. Performance Evaluation

The results are described in this section. In Section 4.1, we discuss the evaluation setting in the UAV infrastructure. In Section 4.2, we present the example-based evaluation results using the attack graph. Finally, in Section 4.3, the comparison result is described.

4.1. Evaluation Setting

In this study, we employed two scenarios, as shown in Figure 3a,b. Scenario 1 is a network topology with six nodes, and the attacker was configured to access the network topology over the Internet, as illustrated in Figure 3a. Furthermore, the GCS provided access to each end node, webserver, fileserver, and UAV. In addition, the UAV was designated as a target node. Scenario 2 consisted of nine nodes for network topology, and this scenario had a firewall that allowed only trusted users to communicate, as illustrated in Figure 3b. In addition, the webserver, fileserver, UAV1, and UAV2 were the end node of the network topology in scenario 2. Furthermore, we set UAV2 as a target node.



(a) Scenario 1 (b) Scenario 2

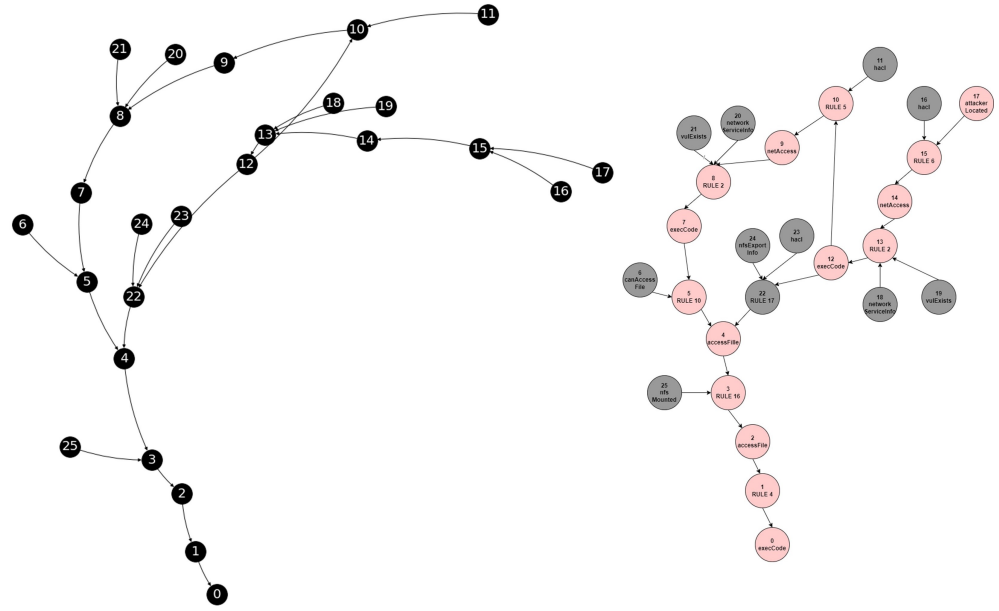
Figure 3. An example of a scenario.

4.2. Example-Based Evaluation Results Using Attack Graph

Additionally, we used a multihost multistage vulnerability analysis tool (MULVAL) to perform COA attacks on UAVs. MULVAL, a logic-based security analyzer, is one of the most extensively used extendable attack graph analysis tools. Depending on the various network topologies, such as in Figure 3a,b, this tool generates a distinct attack graph [36]. The attack graph is a systematic depiction of the vulnerabilities in the system and their relationships generated by the approach of predicting the path the attacker would take to get into the system. In addition, the attack graph represents all network connections. It describes all potential attack paths based on vulnerability information, including CVSS score, the Shodan data used in Mulval, and the network environmental configuration such as firewalls and IDS (intrusion detection system) rules.

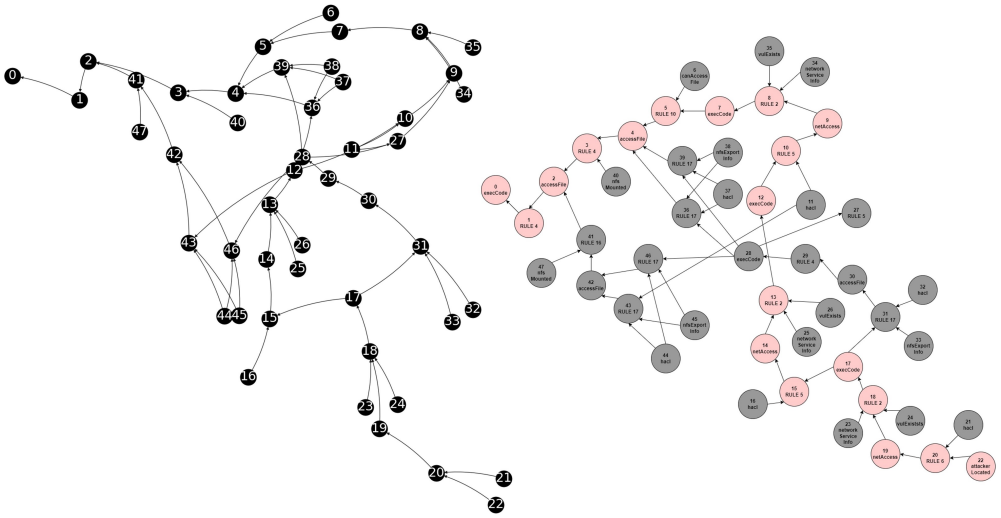
Using MULVAL, we generated attack graphs, as illustrated in Figures 4a and 5 from the UAV-based network environment of scenario 1 and 2, respectively. It can be seen that the complexity of the attack graph increases with the complexity of the scenario. In addition, Tables 2 and 3 show the information of vulnerability including that regarding the network connections or rules of IDs, in scenario 1 and 2, respectively. Furthermore, these tables determine the optimal attack path of the UAV infrastructure. As shown in Tables 2 and 3, the start nodes are the numbers 17 and 22, respectively, indicating the attacker is located on the Internet, while the target node number is 0, indicating the attack is being executed on the UAV. The attack graph has several attack paths from the source node to the target node, and the attacker has no idea which one is the best. As a result, it is critical to identify the optimal attack path in the attack graph.

Using our proposed spatio-attack COA algorithm, we verified the optimal attack path that maximizes the reward in the attack graph of the scalable UAV-based environment. The optimal attack path from the source node to the target node in scenario 1 is in the order of 17 -> 15 -> 14 -> 13 -> 12 -> 10 -> 9 -> 8 -> 7 -> 5 -> 4 -> 3 -> 2 -> 1 -> 0, as shown in Figure 4b. In addition, in scenario 2, the optimal attack path is in the order of 22 -> 20 -> 19 -> 18 -> 17 -> 15 -> 14 -> 13 -> 12 -> 10 -> 9 -> 8 -> 7 -> 5 -> 4 -> 3 -> 2 -> 1 -> 0, as shown in Figure 5b. These attack paths can maximize the reward score between the source and target nodes in each scenario. Therefore, we verified that our proposed algorithms could determine the optimal attack path in various scenarios.



(a) The attack graph (b) The optimal attack path

Figure 4. The result of the execution in scenario 1.

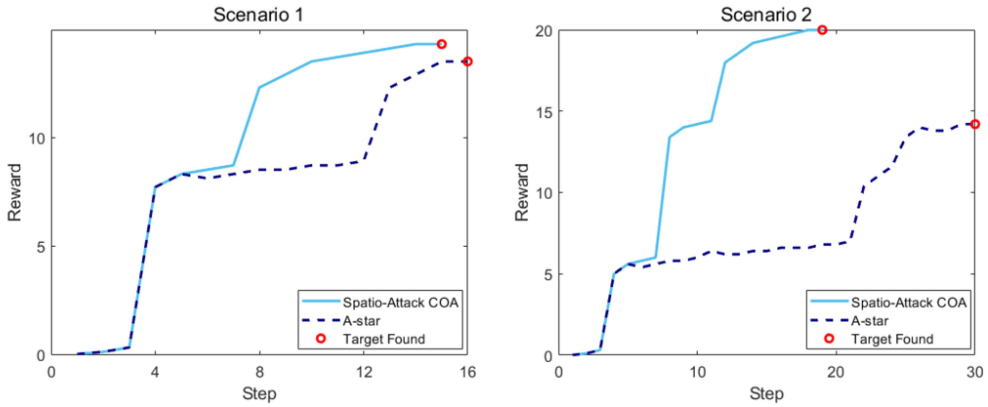


(a) The attack graph (b) The optimal attack path

Figure 5. The result of the execution in scenario 2.

4.3. Comparison Results

In this section, we describe the comparison results between the performance of the spatio-attack COA algorithm and of the A-star search algorithm, which is the base structure of our algorithm. We conducted these comparisons in the two mentioned scenarios and compared the performance using reward and step metrics. The comparison results for the correlation between step and reward in each scenario are shown in Figure 6a,b, respectively for scenario 1 and scenario 2. Note that a short step indicates that the target was detected more quickly, and the attack value increases with reward quantity, representing better performance. In addition, the details on the performance are described in Tables 4–7.



(a) Scenario 1 (b) Scenario 2

Figure 6. The correlation between step and reward in each algorithm.

Table 4. The comparison of step and reward between the spatio-attack COA algorithm and the A-star algorithm in scenario 1.

Algorithm	Reward	Step
Spatio-Attack COA	14.293	15
A-star	13.493	16

Table 5. The details of the correlation between step and reward between the algorithms in scenario 1.

Step	Algorithm			
	Spatio-Attack COA		A-Star	
	Node	Reward	Node	Reward
1	17	0.01	17	0.01
2	15	0.11	15	0.11
3	14	0.31	14	0.31
4	13	7.706	13	7.706
5	12	8.306	12	8.306
6	10	8.506	22	8.106
7	9	8.706	4	8.306
8	8	12.294	3	8.506
9	7	12.894	3	8.506
10	5	13.494	2	8.706
11	4	13.694	2	8.706
12	3	13.894	1	8.906
13	2	14.094	1	12.294
14	1	14.294	1	12.894
15	0	**Target found**	1	13.494
16	-	-	0	**Target found**

Table 6. The comparison of step and reward between the spatio-attack COA algorithm and the A-star algorithm in scenario 2.

Algorithm	Reward	Step
Spatio-Attack COA	19.986	19
A-star	14.198	30

Table 7. The details of the correlation between step and reward between the algorithms in scenario 2.

Step	Algorithm			
	Spatio-Attack COA		A-Star	
	Node	Reward	Node	Reward
1	22	0.01	22	0.01
2	20	0.11	20	0.11
3	19	0.31	19	0.31
4	18	5.002	18	5.002
5	17	5.602	17	5.602
6	15	5.802	31	5.402
7	14	6.002	30	5.602
8	13	13.398	30	5.802
9	12	13.998	30	5.802
10	10	14.198	30	6.002
11	9	14.398	28	6.402
12	8	17.986	28	6.202
13	7	18.586	39	6.202
14	5	19.186	4	6.402
15	4	19.386	4	6.402
16	3	19.586	3	6.602
17	2	19.786	3	6.602
18	1	19.986	3	6.602
19	0	**Target found**	2	6.802
20	-	-	2	6.802
21	-	-	1	7.002
22	-	-	1	10.39
23	-	-	1	10.99
24	-	-	1	11.59
25	-	-	1	13.398
26	-	-	1	13.998
27	-	-	1	13.798
28	-	-	1	13.798
29	-	-	1	14.198
30	-	-	0	**Target found**

4.3.1. Scenario 1

The comparison results of scenario 1 are shown in Figure 6a and Table 4. In the reward, we discovered that the spatio-attack COA algorithm outperforms the A-star algorithm: spatio-Attack COA algorithm (14.293) and A-star algorithm (13.493). In addition, in the step, the spatio-attack COA algorithm requires fewer steps to find the target node than does the A-star algorithm: spatio-attack COA algorithm (14 steps) and A-star algorithm (15 steps). Overall, the spatio-attack COA algorithm devised a more optimal attack path than did the A-star algorithm.

In Table 5, further details of each algorithm's performance are described. As shown in Table 5, the path is identical until the 12 node, with the step being at 5. However, in step 6, each algorithm decides other paths to the target node. Additionally, as shown in the A-star algorithm part of the table, it can be seen that there are duplicate node numbers: steps 8 to 9 (=node 3), steps 10 to 11 (=node 2), and steps 12 to 15 (=node 1). This means that the A-star algorithm searches for the number of cases in path selection, while the spatio-attack COA algorithm does not. Therefore, it can be confirmed that the spatio-attack COA is more efficient than the A-star algorithm in terms of time and cost.

4.3.2. Scenario 2

The comparison results for scenario 2 are presented in Figure 6b and Table 6. In the reward, we discovered that the spatio-attack COA algorithm outperforms the A-star algorithm: spatio-attack COA algorithm (19.986) and A-star algorithm (14.198). In addition,

in the step, the spatio-attack COA algorithm requires fewer steps to find the target node than does the A-star algorithm: spatio-attack COA algorithm (18 steps) and the A-star algorithm (29 steps). Overall, the spatio-attack COA algorithm showed devised a more optimal attack path than did the A-star algorithm.

In Table 7, further details of each algorithm's performance are provided. As shown in Table 7, the path was identical until the node 17, where the step number is 5. However, at step 6, each algorithm decides other paths to the target node. Additionally, as shown in the A-star algorithm part of the table, it can be seen that there are duplicate node numbers: steps 7 to 10 (=node 30), steps 11 to 12 (=node 28), steps 14 to 15 steps (=node 4), steps 16 to 18 (=node 3), steps 19 to 20 (=node 2), and steps 21 to 29 (=node 1). This means that the A-star algorithm searches for the number of cases in path selection, while the spatio-attack COA algorithm does not. Therefore, it can be confirmed that the Spatio-Attack COA is more efficient than the A-star algorithm in terms of time and cost.

4.4. Discussion

In this section, we discuss further details of the experimental results. By comparing the experimental results, we found the optimal attack path for these UAV infrastructure scenarios using the spatio-attack COA algorithm and its benefits.

As shown in the Tables 5 and 7, scenario 1, it can be seen that the spatio-attack COA method significantly outperforms the A-star algorithm in terms of overall reward. In Table 5, scenario 2, the difference in each algorithm's reward is at 0.8, which appears negligible. However, in Table 7, the difference in each algorithm's reward is 5.788, which is significant. Therefore, we could confirm that the reward difference of each algorithm is more significant according to the scales of the scenarios.

In addition, as shown in the Tables 5 and 7, it can be seen that the spatio-attack COA method significantly outperforms the A-star algorithm in terms of overall steps. In Table 5, scenario 1, the difference in each algorithm's is at step is at step 1, which could seem negligible. However, in Table 7, scenario 2, the difference in each algorithm's step is at 11, which is significant. Therefore, we could confirm that the step difference of each algorithm is more significant according to scales of the scenarios. Additionally, if the duplicate nodes and step calculating are calculated, the A-star method's path is shorter overall. However, it can be seen that the spatio-attack COA algorithm made a faster decision by calculating the path while moving forward at each node without duplicating.

Furthermore, since the spatio-attack COA algorithm's structure is built on the A-star algorithm, it has certain advantages. The A-star algorithm's ability to backtrack may help it escape infinite loops. The A-star method stops calculating the heuristic function and backtracks to all nodes constituting the path if a node forming an endless loop is discovered, resulting in the shortest path from the source to the target node [37]. The spatio-attack algorithm, therefore, has the advantage of not having an endless loop.

Based on these benefits, our spatio-attack COA algorithm approach can also be employed for IoT-enabled UAVs. There is also a risk of attacks on UAVs, which supports a variety of IoTs, such as smart farms and virtual environments [38–41]. Therefore, our method can be used to assess security systems and discover their weaknesses.

5. Conclusions and Future Work

Recently, UAVs have been used for various purposes using computer software. UAVs will become more popular, as they provide flexible and robust network services. However, several security vulnerabilities, such as DoS attacks and snooping attacks by malicious hackers, may occur since UAVs interact over the Internet. As a result, identifying an attacker's potential attack behavior in a UAV is essential to maintaining the security of the UAV infrastructure. Therefore, applying the COA technique to UAV infrastructure among cyber security techniques that evaluate system security and identifying vulnerabilities by investigating the network environment are necessary. However, as the existing COA approaches are based on learned data, they cannot be used with scalable networks.

Therefore, in this paper, we proposed and evaluated a novel spatio-attack COA algorithm in the UAV-based scalability network. To handle scalable networks, we developed and implemented the suggested algorithm based on the A-star algorithm, employing optimal heuristic function definitions and intelligent and adaptive search-based approaches. In addition, this algorithm can determine the best attack path to maximize attack value based on the MDP method. Furthermore, we demonstrated that our proposed method performs effectively in two UAV-based network environments.

In future work, we will need to consider time-varying networks because our algorithm now considers only spatially scalable networks. In UAV infrastructure especially, it is essential to consider time-varying networks due to climate environment and connection with other networks. Therefore, we will consider the time-varying networks in UAV infrastructure including Monte Carlo (MC) tree search methods. In addition, our proposed algorithm should be evaluated compared to popular attack COA search methods. The suggested approach will determine if peripheral solutions are more beneficial for the attack COA in scalable and time-varying networks using the MC-based tree search. Moreover, for more practical applications, partially visible networks must be considered. Therefore, as opposed to those that know the whole network topology, we will consider algorithms that emulate actual hackers to determine optimal attack paths based purely on the knowledge of K-hop.

Author Contributions: Conceptualization, S.B.S.; methodology, S.B.S. and D.H.K.; software, S.B.S.; validation, S.B.S.; formal analysis, D.H.K.; investigation, D.H.K.; resources, D.H.K.; data curation, S.B.S.; writing—original draft preparation, S.B.S.; writing—review and editing, S.B.S.; visualization, S.B.S.; supervision, S.B.S. and D.H.K.; project administration, D.H.K.; funding acquisition, D.H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Agency for Defense Development under the contract UI210009XD and the National Research Foundation of Korea (2022R1A2C2004869).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare that there are no conflict of interest.

References

1. Shin, M.; Kim, J.; Levorato, M. Auction-based Charging Scheduling with Deep Learning Framework for Multi-Drone Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 4235–4248. [CrossRef]
2. Sathyamoorthy, D. A Review of Security Threats of Unmanned Aerial Vehicles and Mitigation Steps. *J. Def. Secur. Technol.* **2015**, *6*, 81–97.
3. Sathyamoorthy, D. Key Defence R&D Fields to Develop the National Defence Industry: Focus on C4ISR in Support of Network Centric Operations and Unmanned Vehicles. *Def. S&T Tech. Bull.* **2010**, *3*, 43–60.
4. Bhattacharjee, D. Unmanned Aerial Vehicles and Counter Terrorism Operations. 2015. Available online: http://ssrn.com/abstract=2608969 (accessed on 25 February 2023).
5. Jung, S.; Yun, W.J.; Kim, J.; Kim, J.H. Infrastructure-Assisted Cooperative Multi-UAV Deep Reinforcement Energy Trading Learning for Big-Data Processing. In Proceedings of the International Conference on Information Networking (ICOIN), Jeju Island, Republic of Korea, 13–16 January 2021; pp. 159–162.
6. Jung, S.; Yun, W.J.; Shin, M.; Kim, J.; Kim, J.H. Orchestrated Scheduling and Multi-Agent Deep Reinforcement Learning for Cloud-Assisted Multi-UAV Charging Systems. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5362–5377. [CrossRef]
7. Yun, W.J.; Park, S.; Kim, J.; Shin, M.; Jung, S.; Mohaisen, D.A.; Kim, J.H. Cooperative Multiagent Deep Reinforcement Learning for Reliable Surveillance via Autonomous Multi-UAV Control. *IEEE Trans. Ind. Inform.* **2022**, *18*, 7086–7096. [CrossRef]
8. Jung, S.; Kim, J. Adaptive and Stabilized Real-Time Super-Resolution Control for UAV-Assisted Smart Harbor Surveillance Platforms. *J. Real-Time Image Process* **2021**, *18*, 1815–1825. [CrossRef]
9. Yun, W.J.; Jung, S.; Kim, J.; Kim, J.H. Distributed Deep Reinforcement Learning for Autonomous Aerial eVTOL Mobility in Drone Taxi Applications. *ICT Express* **2021**, *7*, 1–4. [CrossRef]
10. Zanella, A.; Bui, N.; Castellani, A.; Vangelista, L.; Zorzi, M. Internet of Things for Smart Cities. *IEEE Internet Things J.* **2014**, *1*, 22–32. [CrossRef]
11. Saad, M.; Choi, J.; Nyang, D.; Kim, J.; Mohaisen, A. Toward Characterizing Blockchain-Based Cryptocurrencies for Highly Accurate Predictions. *IEEE Syst. J.* **2020**, *14*, 321–332. [CrossRef]
12. Dao, N.N.; Phan, T.V.; Sa'ad, U.; Kim, J.; Bauschert, T.; Do, D.T.; Cho, S. Securing Heterogeneous IoT With Intelligent DDoS Attack Behavior Learning. *IEEE Syst. J.* **2021**, *16*, 1974–1983. [CrossRef]

13. Park, J.; Samarakoon, S.; Elgabli, A.; Kim, J.; Bennis, M.; Kim, S.L.; Debbah, M. Communication-Efficient and Distributed Learning Over Wireless Networks: Principles and Applications. *Proc. IEEE* **2021**, 109, 796–819. [CrossRef]
14. Malik, A.; Kim, J.; Kim, K.S.; Shin, W.Y. A Personalized Preference Learning Framework for Caching in Mobile Networks. *IEEE Trans. Mob. Comput.* **2021**, *20*, 2124–2139. [CrossRef]
15. Kim, D.; Kwon, D.; Park, L.; Kim, J.; Cho, S. Multiscale LSTM-Based Deep Learning for Very-Short-Term Photovoltaic Power Generation Forecasting in Smart City Energy Management. *IEEE Syst. J.* **2021**, *15*, 346–354. [CrossRef]
16. Meteriz, U.; Fazil Yildiran, N.; Kim, J.; Mohaisen, D. Understanding the Potential Risks of Sharing Elevation Information on Fitness Applications. In Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), Singapore, 29 November–1 December 2020; pp. 464–473.
17. Shin, M.; Kim, J. Randomized Adversarial Imitation Learning for Autonomous Driving. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Macao, China, 10–16 August 2019; pp. 4590–4596.
18. Shin, M.; Choi, D.H.; Kim, J. Cooperative Management for PV/ESS-Enabled Electric Vehicle Charging Stations: A Multiagent Deep Reinforcement Learning Approach. *IEEE Trans. Ind. Inform.* **2020**, *16*, 3493–3503. [CrossRef]
19. Testa, A.; Cinque, M.; Coronato, A.; De Pietro, G.; Augusto, J.C. Heuristic strategies for assessing wireless sensor network resiliency: An event-based formal approach. *J. Heuristics* **2015**, *21*, 145–175. [CrossRef]
20. Wang, L.; Chen, Y.; Wang, P.; Yan, Z. Security Threats and Countermeasures of Unmanned Aerial Vehicle Communications. *IEEE Commun. Stand. Mag.* **2021**, *5*, 41–47. [CrossRef]
21. He, D.; Chan, S.; Guizani, M. Communication Security of Unmanned Aerial Vehicles. *IEEE Wirel. Commun.* **2016**, *24*, 134–139. [CrossRef]
22. Shakhatreh, H.; Sawalmeh, A.H.; Al-Fuqaha, A.; Dou, Z.; Almaita, E.; Khalil, I.; Othman, N.S.; Khreishah, A.; Guizani, M. Unmanned Aerial Vehicles: A Survey on Civil Applications and Key Research Challenges. *IEEE Access* **2019**, *7*, 48572–48634. [CrossRef]
23. Gittleson, K. Data-Stealing Snoopy Drone Unveiled at Black Hat. *BBC News*. 2014. Available online: bbc.com/news/technology-26762198 (accessed on 25 February 2023).
24. Gettinger, D. *Domestic Drone Threats*; Bard College Center for the Study of the Drone: Hudson, NY, USA, 2015.
25. Lee, J.; Moon, D.; Kim, I. Technological Trends in Cyber Attack Simulations. *Electron. Telecommun. Trends* **2020**, *35*, 34–48.
26. Nagaraju, V.; Fiondella, L.; Wandji, T. A Survey of Fault and Attack Tree Modeling and Analysis for Cyber Risk Management. In Proceedings of the IEEE International Symposium on Technologies for Homeland Security (HST), Boston, MA, USA, 25–26 April 2017; pp. 1–6.
27. Ghosh, N.; Ghosh, S.K. A Planner-based Approach to Generate and Analyze Minimal Attack Graph. *Appl. Intell.* **2012**, *36*, 369–390. [CrossRef]
28. Liang, X.; Xiao, Y. Game Theory for Network Security. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 472–486. [CrossRef]
29. Zhou, S.; Liu, J.; Hou, D.; Zhong, X.; Zhang, Y. Autonomous Penetration Testing Based on Improved Deep Q-Network. *Appl. Sci.* **2021**, *11*, 8823. [CrossRef]
30. Hong, J.; Kim, D.S. *Harms: Hierarchical Attack Representation Models for Network Security Analysis*; Edith Cowan University: Perth, Australia, 2012.
31. Schwartz, J.; Kurniawati, H.; El-Mahassni, E. POMDP+ Information-Decay: Incorporating Defender's Behaviour in Autonomous Penetration Testing. In Proceedings of the International Conference on Automated Planning and Scheduling, Nancy, France, 26–20 October 2020; Volume 30, pp. 235–243.
32. Sutton, R.S. Introduction: The Challenge of Reinforcement Learning. In *Reinforcement Learning*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 1–3.
33. AlShawi, I.S.; Yan, L.; Pan, W.; Luo, B. Lifetime Enhancement in Wireless Sensor Networks Using Fuzzy Approach and A-Star Algorithm. *IEEE Sens. J.* **2012**, *12*, 3010–3018. [CrossRef]
34. Mell, P.; Scarfone, K.; Romanosky, S. A Complete Guide to the Common Vulnerability Scoring System Version 2.0. In *Forum of Incident Response and Security Teams (FIRST)*; Forum of Incident Response and Security Teams (FIRST): Cary, NC, USA, 2007.
35. Hu, Z.; Beuran, R.; Tan, Y. Automated Penetration Testing using Deep Reinforcement Learning. In Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 16–18 June 2020; pp. 2–10.
36. Ou, X.; Govindavajhala, S.; Appel, A.W. MulVAL: A Logic-based Network Security Analyzer. In Proceedings of the USENIX Security Symposium, Baltimore, MD, USA, 31 July–5 August 2005; Volume 8, pp. 113–128
37. Sharma, S.; Srijan, S.; Vidhya, J. Parallelizing Bidirectional A* Algorithm. In *Intelligent Systems and Computer Technology*; Advances in Parallel Computing; IOS Press: Amsterdam, The Netherlands, 2020; Volume 37, pp. 558–562. [CrossRef]
38. Boursianis, A.D.; Papadopoulou, M.S.; Diamantoulakis, P.; Liopa-Tsakalidi, A.; Barouchas, P.; Salahas, G.; Karagiannidis, G.; Wan, S.; Goudos, S.K. Internet of things (IoT) and Agricultural Unmanned Aerial Vehicles (UAVs) in Smart Farming: A Comprehensive Review. *Internet Things* **2022**, *18*, 100187. [CrossRef]
39. Islam, N.; Rashid, M.M.; Pasandideh, F.; Ray, B.; Moore, S.; Kadel, R. A Review of Applications and Communication Technologies for Internet of Things (IoT) and Unmanned Aerial Vehicle (UAV) Based Sustainable Smart Farming. *Sustainability* **2021**, *13*, 1821. [CrossRef]

40. Israr, A.; Abro, G.E.M.; Sadiq Ali Khan, M.; Farhan, M.; Bin Mohd Zulkifli, S.u.A. Internet of Things (IoT)-Enabled Unmanned Aerial Vehicles for the Inspection of Construction Sites: A Vision and Future Directions. *Math. Probl. Eng.* **2021**, 2021, 9931112. [CrossRef]
41. Li, W.; Cai, S.; Zheng, H. Research on Multi-task Cruise Path Planning of UAV Based on COA Optimization Algorithm. In Proceedings of the 3rd Asia-Pacific Conference on Image Processing, Electronics and Computers, Dalian, China, 14–16 April 2022; pp. 662–666.