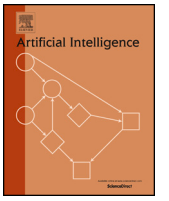
Artificial Intelligence 326 (2024) 104032

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Distributed web hacking by adaptive consensus-based reinforcement learning

Nemanja Ilić [a,b], Dejan Dašić [a,c,*], Miljan Vučetić [a,c], Aleksej Makarov [a], Ranko Petrović [a]

[a] *Vlatacom Institute of High Technologies Ltd., Belgrade, Serbia*
[b] *College of Applied Technical Sciences, Department of Information Technologies, Kruševac, Serbia*
[c] *Singidunum University, Belgrade, Serbia*

ARTICLE INFO

Keywords:
Distributed reinforcement learning
Multi-agent system
Adaptive consensus-based algorithm
Distributed Q-learning
Ethical web hacking
Penetration testing
Capture the flag

ABSTRACT

In this paper, we propose a novel adaptive consensus-based learning algorithm for automated and distributed web hacking. We aim to assist ethical hackers in conducting legitimate penetration testing and improving web security by identifying system vulnerabilities at an early stage. Ethical hacking is modeled as a capture-the-flag style task addressed within a distributed reinforcement learning framework. To achieve our goal, we employ interconnected intelligent agents that interact with their copies of the environment simultaneously to reach the target. They perform local information processing to optimize their policies and exchange information with neighboring agents. We propose a novel adaptive consensus scheme for inter-agent communications, which enables the agents to efficiently share network-wide information in a decentralized manner. The scheme dynamically adjusts its weights based on heuristics, involving both recency and frequency metrics of actions selected at a given state by an individual agent, similar to eligibility traces. We extensively analyze the convergence properties of our algorithm and introduce a new communication scheme design. We demonstrate that this design ensures the fastest convergence to the desired asymptotic values under the general setting of asymmetric communication topologies. Additionally, we provide a comprehensive review of the current state of the field and propose a web agent model with improved scalability compared to existing solutions. Numerical simulations are conducted to illustrate the key characteristics of our algorithm. The results demonstrate that it outperforms both non-cooperative and average consensus schemes. Moreover, our algorithm significantly reduces hacking times when compared to baseline algorithms that rely on more complex models. These findings offer valuable insights to system security administrators, enabling them to address identified shortcomings and vulnerabilities effectively.

1. Introduction

Digital transformation and the Internet pervasiveness have increased productivity and efficiency but exposed business processes, critical data and infrastructures to unprecedented security and financial risks. In 2020, the average business cost of a cyberattack

* Corresponding author at: Vlatacom Institute of High Technologies Ltd., Belgrade, Serbia.
E-mail address: dejan.dasic@vlatacom.com (D. Dašić).

https://doi.org/10.1016/j.artint.2023.104032
Received 15 August 2022; Received in revised form 25 July 2023; Accepted 12 October 2023

Available online 24 October 2023
0004-3702/© 2023 Elsevier B.V. All rights reserved.

was $3.86 million [1]. Cyberattacks were projected to hit $6 trillion in annual loss in 2021 [2], which is around 6% of the global annual gross domestic product [3]. State espionage is the second major cause of cyberattacks [4]. Public security and human lives are threatened by malicious activities conducted through IT infrastructure, as witnessed by riots due to the unavailability of Estonian e-government and banking services in 2007 [5], attack on Iranian nuclear plant in 2010 [6], massive breaches of the US voter database in 2015 [7], or the EU communication network eavesdropping between 2015 and 2018 [8].

Web applications are nowadays omnipresent in human-to-machine (H2M) communications. Web applications are server-side programs such as contact forms, classified advertisement webpages, webmail, instant messaging, social networks, forums, search engines, automatic translators, banking or retail websites, accessed by users through a web browser. They typically exchange data between users and a web server through communication protocols. If not properly secured, by using digital certificate-based authentication, encryption, secured session cookies, parametrized queries, firewalls, timely patching and audits, the exchanged or stored information can be intercepted, stolen or tampered. A recent website security statistics report of White Hat found that at least 50 percent of web applications had one or more serious vulnerabilities [9].

Organizational flaws and computer network vulnerabilities are explored by ethical hackers, whose findings can be used to prevent threats from unknown attackers in the future. The degree to which a hacker can pierce the defense of a system is often referred to as penetration. More than 80% of all malware attacks are performed by autonomous bots [10]. The focus of this paper is on the use of such bots, i.e., intelligent agents, in ethical hacking. They can also be used in malicious attacks, but our scope and goals are, of course, exclusively within the domain of ethical penetration testing.

In this paper, we model agent interaction with the system via the Reinforcement Learning (RL) [11] framework. Since time plays a vital role in hacking tasks, we dispatch multiple cooperative agents operating within the distributed RL context, which would hopefully allow for faster solutions. The agents interact with multiple copies of the environment in parallel, trying to reach the target by: 1) processing local information (learning from interaction experience), and 2) communicating with each other. Local processing is focused on agent policy optimization, aimed at providing agents with the optimal choice of actions at each state they encounter. Inter-agent communication is aimed at providing all agents with viable network-wide information.

The setting we adopt in this paper is similar to the one described in seminal papers from the field of distributed RL [12–14]. It involves multiple agents, referred to as actors-learners, operating in parallel. However, we propose a scalable and robust fault-tolerant solution by empowering all agents with global learning results in a completely distributed and decentralized manner. For comparison, we also simulate the centralized communication scheme and include the results in the figures.

To achieve a fast algorithm, we prioritize the learning results of agents that possess relevant information. This prioritization is not achieved through a global prioritized entity, such as a prioritized experience replay memory [15], but rather through the design of the inter-agent communication scheme. For simplicity, the communication protocol is applied to the agents' local processing results in an ad-hoc manner. It is not incorporated as a part of the RL formalism [16], nor modeled within the framework of Markov games [17], which considers joint actions of multiple agents interacting within a single environment [18,19].

With respect to the inter-agent communications, we concentrate on consensus algorithms, that have demonstrated their effectiveness in facilitating efficient coordination among multiple agents [20,21]. The objective of this paper is to present an innovative adaptive consensus-based distributed RL scheme aimed at assisting ethical hackers in conducting legitimate penetration testing in an efficient and scalable manner. Our focus is on achieving fast and reliable results. We extend the web agent model described in [22] to the distributed RL context and propose a modified version that enables scalability to large networks. For local processing, we employ the well-known Q-learning algorithm [11]. We do not impose any restrictions on the actions selection strategy as in [23]. The consensus schemes we consider can also be applied to function approximation methods, as demonstrated in [24].

According to consensus-based approaches, we assume that each agent communicates with only a subset of other agents, referred to as neighbors. By appropriately designing these neighbor-based information exchanges, the desired global results can be obtained after multiple propagations through the network. Our proposed adaptive consensus algorithm for inter-agent communication assigns greater weight to agents with higher-quality local information. We utilize both frequency and recency metrics to evaluate the actions chosen by an individual agent at a given state. These metrics are incorporated into a unified heuristic, similar to eligibility traces [11], to determine the weight of the communicated variables. This adaptive scheme represents the main contribution of our paper within the context of consensus-based distributed RL. With the introduction of this proposed adaptive scheme, the collective knowledge of the entire network is dominated by the nodes possessing higher-quality information, as in [25,26]. Simple averaging of local information, as enabled by different strategies [24,27–29], does not facilitate reasonably fast solutions. To the best of our knowledge, consensus-based distributed RL schemes specifically tailored for ethical hacking have not been previously proposed.

This paper thoroughly examines the convergence properties of the consensus algorithm employed and proposes a communication scheme that guarantees the fastest convergence to the desired asymptotic values. The technical analysis builds upon classical results [30], incorporating a novel contribution that extends the findings of [30] to encompass the general case of asymmetric communication topologies. In addition to these technical aspects, the paper offers a comprehensive review of the current state of the field.

We also conduct extensive numerical simulations, encompassing different models, algorithms and underlying network topologies. The simulations confirm the improved efficacy and scalability properties of the proposed model, compared to the original model presented in [22]. Furthermore, the benefits of adapting the consensus scheme are clearly demonstrated. The proposed algorithm yields significant reductions in hacking times, consistently across different topologies, and also under randomized communication protocols. Beside outperforming non-cooperative and average consensus schemes, the adaptive scheme also outperforms baseline algorithms [31] that assume more complex models.

The structure of the paper is organized as follows. Section 2 provides an overview of the tools and methodologies utilized in penetration testing. Section 3 covers web hacking concepts and various types of web vulnerabilities. This section also discusses the con-

cepts employed in our approach to penetration testing, such as the "Capture the Flag" challenge, RL fundamentals, and the adaptive consensus scheme. In Section 4 we describe the design of the agent model and introduce a novel consensus-based learning algorithm that enables rapid detection of vulnerabilities in web applications. In Section 5, we present the experimental setup and the achieved results. Section 6 summarizes the content presented in the preceding sections and outlines potential directions for future research.

2. Related work

Development of artificial intelligence (AI) tools for the needs of cyber-security is a very wide and attractive field of research [32]. The design and implementation of intelligent and autonomous agents for tackling various problems [33] is a mere fragment of this challenging arena; one in which the authors feel the proposed algorithm is a humble contribution.

Ever since the introduction of web applications, even simple websites, ways of protecting them from unwanted intruders have been the topic of consideration. Research of literature shows several different formalizations of penetration testing problem.

Off-the-shelf web application security audit tools are often limited by either human resources or target environment changes. Both of these factors make the security audit processes time consuming and/or error-prone [34]. A list of such tools, used for the purpose of penetration testing, may include software like nmap - a popular port scanner [35], Wireshark - a well-known network protocol analyzer [36], John the Ripper - password cracker [37], Burp Suite - web vulnerability scanner [38], etc. There are also penetration testing frameworks, like Metasploit [39], and even operating systems dedicated to penetration testing, such as Kali Linux [40], which readily ships with most of the tools mentioned.

Early research on automation of penetration testing has been rooted in classical planning. In [41], plans of attack against a web-based document management system were demonstrated, which encompassed the sequences of exploits necessary to reach the goal of an attacker corresponding to the possible courses of action of the defender. In [42], an attack model developed in planning domain definition language (PDDL) was implemented within an automatic penetration tester as a planning module capable of receiving the information collected through hacking and taking the appropriate action. As opposed to deterministic planning approaches, the work presented in [43] shows that attack planning based on probabilistic actions can efficiently test the resilience of large-scale IT networks, by modeling the probabilities of the outcomes of possible actions.

Very closely related to the planning based formulation of penetration testing are the approaches based on attack graphs, originally introduced by authors in [44]. Work presented in [45] considers the penetration testing attack planning problem in terms of partially observable Markov decision processes (POMDP), allowing the authors to use this well-known tool to model information gathering achieved by scanning actions as an integral part of the penetration problem, thus combining scanning with exploits, i.e., hacking actions that actually modify the system. Building, in part, on [45] and [46] and in order to model the behavior of the defender, to either analyze or alter its network, the authors in [47] include an Information-decay parameter in the POMDP-based autonomous penetration testing framework, thus accounting for all of the causes of uncertainty.

Another notable take on the problem of penetration testing is the application of game theory. In [48] penetration testing was modeled as a method for increasing the system predictability through collecting the system information. It was shown that this kind of probing generally increases the efficiency of investment made in security, in terms of reducing the risk and the total cost of managing the system. One of the approaches to penetration testing based on application of game theory are cyber security games. These concepts were used in [49], where authors present a cyber security gaming framework, modeling both the assailant and the defender having in mind the readily available exploit kits, with the idea to shape the environment-specific cyber defense policy. Researchers in [50] employ Stackelberg planning in order to provide a basis for a holistic mitigation strategy based on minimizing the maximal attacker's success. Work presented in [51] notes analogies between establishing cyber wargaming strategies and the approach used by armed forces called Course of Action (COA) generation. Authors in [52] explore Bayesian game as the basis for a framework to model interactions between attackers trying to disguise their activities and investigators who aim to find them out.

Another area of research through which the problem of penetration testing has seen much treatise lately is reinforcement learning (RL). Work presented in [53–56] establishes RL as a viable candidate for both planning and execution tasks of the penetration testing schemes. One of the challenges in attack planning is lack of knowledge on the structure of network under attack. To address this issue, researchers in [57] propose a network information gain based attack planning algorithm for penetration testing, in which RL model guides the discovery of the attack paths based on a cumulative network information gain. Due to the increasing size and complexity of web applications and underlying networks, Deep RL (DRL) is introduced to try to manage the "curse of dimensionality" of the problem, by having neural network structures approximate the values of functions required for RL algorithm calculations. Research that aims in this direction in the field of penetration testing can be seen in [58,59]. Introduction of actor-critic algorithms to DRL [13,60] is shown to be a promising approach to the penetration testing problem when dealing with very high number of hosts [61]. Even though there are works which question the effectiveness and performance of state-of-the-art deep learning techniques in real-world vulnerability prediction scenarios [62], DRL with actor-critic algorithms has seen implementation in some of the most used publicly available tools for penetration testing. One of such tools is the Deep Exploit [63], which uses distributed multi-agent RL with A3C (asynchronous advantage actor critic) [64].

The investigated subject has garnered a large number of papers. Still, a thorough examination of the literature, including a meta-analysis concentrating on AI in penetration testing and vulnerability assessment [65], reveals that only a limited number of papers, precisely 16, published between 2010 and 2018, meet the specific research criteria. These criteria involve a description of the machine learning or artificial intelligence algorithms or application, accompanied by an empirical study.

An overview of the methodologies applied to tackle the issue of penetration testing in the surveyed body of knowledge is given in Table 1. With respect to the existing approaches that use RL and DRL, the proposed algorithm is expected to be more efficient

Table 1
Overview of related work.

Authors	Methodology	Accent
M. Boddy et al. [41]	Classical planning	Generation of attack plans for a simple but realistic web-based document control system
J. Lucángeli Obes and C. Sarraute and G. Richarte [42]	Classical planning, attack graphs	Representation of an attack model made in PDDL and an integration of a planner into a penetration testing tool
C. Sarraute, G. Richarte and J. Lucángeli Obes, Jorge [43]	Classical planning	Attack planning problem is considered in the context of regular automated penetration testing, as in existing commercially available tools; the uncertainty about the results of the actions is modeled as a probability of success of each action
C. Phillips and L. P. Swiler [44]	Attack graphs	Introduction of a graph-based approach to network vulnerability analysis
C. Sarraute, O. Buffet and J. Hoffmann [45]	Attack graphs, POMDP	Attack planing problem is modeled in terms of partially observable Markov decision processes (POMDP)
C. Sarraute, O. Buffet and J. Hoffmann [46]	POMDP	POMDPs are used to find good attacks on individual machines and a composition of these constitutes an attack on the network as a whole
J. Schwartz, H. Kurniawati and E. El-Mahassni [47]	POMDP	POMDP-based autonomous penetration testing framework models the defender's actions through an information decay factor, thus taking into consideration all the causes of uncertainty in the attack planning problem
R. Böhme and M. Félegyházi [48]	Game theory, security game	Penetration testing considered as an information gathering option in order to reduce uncertainty in a discrete time security game
J. Robertson et al. [49]	Game theory	System-specific defender policy recommendations extracted from a data-driven security game framework modeling both the attacker and the defender
P. Speicher et al. [50]	Game theory	Analysis of automated mitigation of attacks in a network based on Stackelberg planning
E.J.M. Colbert, A. Kott and L. P. Knachel [51]	Game theory	Comparison of cyber wargames to Course of Action military practice
A. Nisioti et al. [52]	Game theory	Bayesian game of incomplete information employed to model interactions between a cyber forensic Investigator and strategic Attackers that deploy anti-forensic techniques to conceal their activities
M. C. Ghanem and T. M. Chen [53]	Reinforcement learning	POMDP model and solvers used in a RL scheme to design the planning phase of an intelligent penetration testing approach to enable systematic testing without employing human resources
M. C. Ghanem and T. M. Chen [54]	Reinforcement learning	Intelligent Automated Penetration Testing System (IAPTS) enabling information capture, experience learning as well as reproduction of tests in similar testing cases related to different network infrastructures
M. C. Ghanem and T. M. Chen and E. G. Nepomuceno [55]	Reinforcement learning	Major difficulty when solving large POMDPs resulting from large networks in [54] overcome by representing networks hierarchically as a group of clusters and treating each cluster separately
J. Schwartz and H. Kurniawati [56]	Reinforcement learning	Fast, light-weight and open-source network attack simulator, obtained by observing PT as a Markov Decision Process (MDP), effectively applicable to smaller networks and numbers of actions
T. Zhou at al. [57]	Reinforcement learning	PT attack planning algorithm, in which network information gain is the basis for discovery of the attack paths, proposed to tackle the issue of lack of prior knowledge of the environment infrastructure
S. Zhou et al. [58]	Deep reinforcement learning	An improved deep Q-network (DQN) proposed to address the sparse reward problem in large scale PT scenarios
Z. Hu, R. Beuran and Y. Tan [59]	Deep reinforcement learning	A realistic network topology built by collecting relevant server data; vulnerability analysis employed to generate a corresponding attack tree. Deep Q-Learning used afterward to discover the attack path easiest to exploit
H. Nguyen et al. [61]	Double-agent reinforcement learning	Double agent architecture (DAA), using a popular advantage actor-critic (A2C) RL algorithm, proposed for PT in large networks.

due to its simplicity - using POMDP models [53,54] does not scale well with network size [57,66]; on the other hand, DRL models [58,59,64] require more data and training steps to converge. The algorithm's disadvantages caused by its simplistic design are to be compensated by the adopted parallel multi-agent setting coupled with the proposed communication scheme. Other approaches, such as dividing the original problem into subsets to be treated separately [55], or dividing the agent itself into two parts [61], represent also feasible solutions, with a potential to be further enhanced by combining them with the proposed adaptive consensus algorithm.

3. Theoretical background

3.1. Web hacking

Web hacking refers to exploitation and compromising of web-based applications. A web application architecture is realized through a client – server model with HTTP as communication protocol. Using the HTTP protocol, a client sends a request to a server to either deliver a content or record it (using "GET" and "POST" methods, respectively), while the server sends a response message that contains objects back to the client. The object can be a base HTML file either referencing other objects or containing embedded programs written in a scripting language, typically PHP or JavaScript. As it is shown in Fig. 1, all the web application elements can

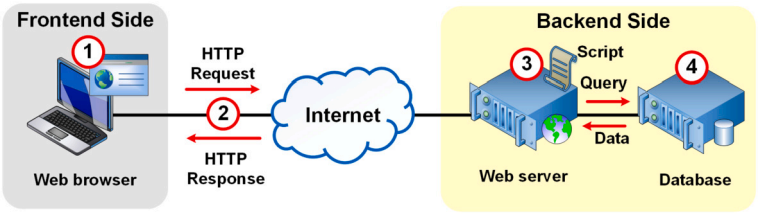


Fig. 1. Components exposed to web-related attacks.

be targets of potential attacks: the web server (position 3 in Fig. 1) and database (position 4 in Fig. 1) at the backend side, the web browser (position 1 in Fig. 1) at the client side and even the network traffic (position 2 in Fig. 1) between them.

The concept of a "web application attack" indicates a violation of some of the three principles of information security: confidentiality, integrity, and availability (the CIA Triad). The confidentiality principle refers to the ability of a system to prevent unauthorized access to information. The principle of integrity implies that data cannot be unnoticeably modified. The availability principle means that information must be accessible to authorized users at all times. Web hacking attacks can be categorized in different ways. They are generally classified into four major groups [67]:

- interception attacks - allowing unauthorized users to access confidential information (eavesdropping and wiretapping),
- interruption attacks - causing a system to become temporarily or permanently unavailable to authorized users (by deliberately overloading a server to the point of unresponsiveness or redirecting requests to invalid destinations),
- modification attacks - information tampering (data substitution, insertion or deletion),
- fabrication attacks - generating fake and potentially harmful information or processes (SQL or route injection).

Additionally, passive and active attacks can be distinguished [68]. In passive attacks, only an observation of a system is performed, without any impact on the system resources and the data content. Active attacks, on the other hand, are attempts to exploit the system resources and alter the data. While passive attacks violate confidentiality, active attacks impair availability and integrity. Interruption, modification and fabrication attacks are active attacks, while interception attacks are considered as passive attacks.

3.2. Types of web vulnerabilities

Many authors [69–75] cite the Open Web Application Security Project [76] and its quadrennially updated list of top ten security risks as the cybersecurity industry's benchmark list for classifying attacks [77]. This section describes the most common web application hacking attacks:

- SQL injection - attack wherein malicious SQL queries are entered in the input field of a web application, in order to intrude or alter the database (active attack that targets the backend of a web application and affects the data integrity and confidentiality),
- Cross-site scripting (XSS) – attack wherein a malicious script is injected into a webpage, forcing the victim's browser to run the script while downloading the page (active attack that compromises confidentiality and integrity, potentially at both client and server sides),
- SYN flood - A distributed denial of service (DDoS) attack in which the initial connection request (SYN) packets are repeatedly sent without sending acknowledgment (ACK) packets to finish the handshake process of a transfer control protocol (TCP) connection (active attack that targets the backend of a web application and affects the data availability),
- HTTP flood – DDoS attack that relies on a large number of legitimate HTTP GET or POST requests to a web server (active attack that targets the backend of a web application and affects the data availability),
- Eavesdropping – man in the middle attack that relies on intercepting the data in transit but without changing the content (passive attack that targets the communication component of a web application and affects the data confidentiality),
- Session hijacking – attack that consists of the exploitation of the web session control mechanism, compromising the session token by session sniffing, or injecting malicious code (active, and in some cases - passive attack that targets all components of a web application and affects confidentiality),
- WEB parameter tempering – man in the middle attack that relies on modifying the intercepted data such as user credentials and permissions usually stored in cookies, URL query strings (active attack that targets the communication component of a web application and affects the data integrity),
- Broken authentication attack – brute force or dictionary attack in which the vulnerabilities of the authentication procedure are exploited in order to get the credentials of other authorized users (active attack that targets backend side of a web application and affects the data confidentiality),
- Sensitive data exposure – attack in which web application unintentionally reveals sensitive information such as source code, database tables, IP addresses (passive attack that targets the backend side of a web application and affects the data integrity and confidentiality).

More detailed classification of the aforementioned web application attacks is given in Table 2.

Table 2
Classification of web application attacks.

	Type of Attack						CIA Triad			Target		
	Interception	Interruption	Modification	Fabrication	Passive	Active	Confidentiality	Integrity	Availability	Client	Communication	Backend
SQL injection				•		•	•	•				•
Cross-site scripting (XSS)				•		•	•	•		•		•
SYN flood		•				•			•			•
HTTP flood		•				•			•			•
Eavesdropping	•				•		•				•	
Session hijacking	•			•	•	•	•			•	•	•
Web parameter tempering			•			•		•			•	
Broken authentication attack				•		•	•					•
Sensitive data exposure	•				•		•	•				•

3.3. Capture the flag

The idea of breaking a cyber-defense of a system for the show is akin to gaming, where players are traditionally labeled as the red (the attackers) and the blue (the defenders, i.e., the system engineers, developers and administrators who conceived and implemented the defense). When an attack is detected, some security measures can be taken by the system administrator to protect the system (e.g., denying access to some IP addresses, closing a port or even pulling the system off the network). This is similar to the outdoor game "Capture the flag", where attackers trying to steal a special item (a "flag") are temporarily neutralized ("tagged") if intercepted by defenders. In cyber security, Capture the Flag (CTF) is a penetration testing exercise in which "flags" are purposefully hidden in programs, databases or websites. Flags can be alphanumeric strings, credentials, files or links. Their capture by a tester is considered to be the proof of penetration. The interest of governments in cyber wars leads to automated software agents, which should perform penetration in milliseconds, before the human system administrator becomes aware of the attack. The DARPA's Cyber Grand Challenge in 2016 introduced such agents – autonomous bots. During this event, the competing teams used the bots they developed (Cyber Reasoning Systems - CRS) to automatically identify software flaws, and scan a network to identify affected hosts. The software agents were challenged to find and instantly patch flawed code that was vulnerable to being hacked, and find their opponents' vulnerabilities [78]. Unlike human attackers, bots can perform only a limited predefined set of actions to probe a system. As systems (e.g. networks) are generally composed of a plurality of nodes (servers, files, web pages), the bots should be endowed with a formalized way of learning the system structure (nodes and connections between them). Finally, they should decide which action is the most appropriate at a given node. This decision-making process in the context of an unknown structure and the hidden flag requires a learning algorithm, with clearly defined local and global objectives (e.g., selecting the next node to act upon and finding the shortest path to the flag), as well as related rewards and penalties [79].

3.4. Reinforcement learning basics

Reinforcement learning (RL) is a trial-and-error approach to training agents, based on their interactions with a possibly unknown environment. An action performed by an agent changes the current state of the environment and gets rewarded according to the degree of desirability of the accomplished state transition. A typical objective for an agent is to maximize the long-term average of rewards for its actions, tending towards the adoption of an optimal policy of behavior [11].

In RL, environments are most commonly modeled as Markov decision processes (MDP). A discrete MDP can be represented by a quadruple $(S, \mathcal{A}, R, T)$, where S is a set of available states, $\mathcal{A}$ is a set of available actions, $R : S \times \mathcal{A} \to R$ is a reward function producing either a dedicated or an average reward (in case of random rewards) for an action performed at a certain state and $T : S \times \mathcal{A} \to S$ is the state transition function.

A policy function $\pi : S \to \mathcal{A}$ provides a feasible action a to be performed for a system at a state s (at a time step t). This function can also be expressed as a probability distribution over a set of actions and states $\pi : S \times \mathcal{A} \to [0, 1]$ (randomized policy). For a given policy function, RL guides the agent to find the optimal policy π^* which maximizes a function of the rewards received for the actions taken over time. This learning goal can thus be expressed as:

$$\text{find } \pi^* \text{ maximizing } E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right], \tag{1}$$

where E_π denotes mathematical expectation of the realizations assuming the policy function π, and $0 \leq \gamma \leq 1$ is the factor of discount for future rewards.

The optimal policy can be computed using either the state-value or the action-value function. The state-value function $v_t^\pi(s)$ represents the expected reward for abiding to policy π starting from state s (at time step t). The function that we will be using to calculate the optimal policy of an agent is the action-value function $Q_t^\pi(s, a)$, which represents the expected reward attained when an

agent executes an action a from the state s (at a time step t) and then continues with the policy π. The optimal action-value function, associated to π^*, can be calculated as [11]:

$$Q_t^{\pi^*}(s,a) = \sum_{s' \in S} T_t(s,a,s') \left(R_t(s,a) + \gamma \max_{a'} Q_t^{\pi^*}(s',a') \right). \tag{2}$$

In case more (than one) agents participate in a setting, cooperating on a joint task, than the value of an action taken at a certain state can, in some way, be shared, thus contributing to a possibly faster discovery on an optimal policy of behavior of all agents. This is one of the ideas behind the motivation of introducing distributed multiagent RL scheme.

3.5. Adaptive consensus-based distributed RL

The goal of this paper is to propose a novel adaptive consensus-based distributed algorithm for the purpose of penetration testing of web applications. We assume that the networked agents are aimed at solving a common task (capturing the flag, e.g. accessing the sensitive information located at the web server) which involves the same underlying MDP process. In this regard, the agents interact with their own copies of the environment independently (i.e. they do not interact with each other through the MDP so that the induced MDP transitions are independent) and in parallel. The task the agents are solving is to find the optimal action-value function, which directly implies the optimal policy (from which the optimal sequence of actions for capturing the flag can be obtained). It is also assumed that the agents can exchange local processing results with their neighbors via the consensus communication scheme. We propose which information should be communicated and how should it be propagated through a given network in order to provide all the agents with valid and reliable results in a fastest possible manner. The goal of the proposed adaptive mechanism is to grant agents having better "quality" information with, roughly speaking, higher "voting" rights.

In general, consensus schemes have been used within this and similar contexts because of their high scalability, since they do not require all-to-all but only neighbor-based communications, and also because of their high robustness to node failures, since no central or fusion entities representing highly vulnerable points in the network are assumed. Consensus algorithms have proven their value as tools for agent coordination (which can efficiently exploit different agents covering possibly complementary parts of the state space), algorithm parallelization (enabling faster convergence) and noise reduction (due to the resulting averaging of the local processing results) [20,21]. Herein, we shall utilize all these properties. The distinction between the agents is made by assuming different starting positions in the underlying MDP (various web pages at particular web applications, etc.), but other settings can also be used (e.g. different behavior policies connected to different agents [13,29]). Our scheme assumes that the agents exchange filtered information, which offers several advantages compared to communicating raw data. These benefits include improved noise reduction, efficient bandwidth usage, faster convergence, and ultimately, privacy preservation. The ability to maintain privacy can be a valuable feature in the specific application scenario being adopted.

4. Distributed web hacking

4.1. Agent model design

We assume that the considered problem of web hacking can be specified by the "Capture the Flag" formulation discussed in Subsection 3.3 and that the agents' interaction with the environment can be modeled within the RL framework described in Subsection 3.4. The underlying MDP is obtained by modeling the logic of the target webserver, represented by a collection of generic objects [22]. These objects represent entities of interest (such as e.g. files or ports, or database authentications credentials) which can be targeted by attacker's actions. The adopted setting allows for the decomposition and modularity of a target system and agent design. It also makes possible to define instances of target webservers at different levels of abstraction. In [22], 7 such levels of abstraction have been proposed, assuming a single attacker (agent).

In this paper, we build upon the implementation of one level of abstraction from [22], namely the so-called Level1 web agent model, where the task is to design a web agent that would successfully search a network (an abstraction of a simple website) composed of a set of nodes (web pages - static HTML files) and a set of links (web links) for a node that contains the flag. The flag itself represents an abstraction of the relevant information the agent is interested in. In an actual scenario, in case of an information disclosure attack, a flag could mark e.g. a configuration file. Other types of web vulnerabilities (e.g. SQL injection, web parameter tempering, etc.) can be modeled with higher level web models from [22]. In case of a XSS attack, e.g., the flag could mark a web page accessible only indirectly by redirection.

In the Level1 web agent model used, we assume that the agent can have access to all nodes and can only perform actions related to the discovered nodes on a website. For simplicity, we assume a fixed node network topology and flag location, although this assumption is not necessary, as the proposed algorithm has the potential to overcome this limitation [29]. While the transitions in equation (2) are defined as stochastic, following the theoretical setting in [22], the actual transitions in the agent model used are deterministic, as implied by the practical implementation of the model from [22].

The model assumes that for each node, the agent can take two possible actions: *Read* (to retrieve a list of linked nodes) and *Search* (to determine if the flag is present, returning a Boolean value). Table 3 provides a summary of the action space in the described model (referred to as "Large"). In the original model [22], there was a third action, *None*, which had no effect other than receiving a reward. All actions except the *Search* action on the node with the flag result in a reward of -1, while the *Search* action on the flag-carrying node yields a reward of 100. Consequently, there are four possible states for each node, as shown in Table 4.

Table 3
Action spaces of the two used agent models.

Model	Action	Observation	Reward	Action space size
Large	*Read (node i)*	List of linked nodes	-1	$3N$
	Search (node i)	False (flag not in node i)	-1	
		True (flag in node i) *Ends episode	100	
	None (node i)	None	-1	
Small	*Read (current node)*	List of linked nodes	-1	$2+N$
	Search (current node)	False (flag not in node i)	-1	
		True (flag in node i) *Ends episode	100	
	Switch (to node i)	None	-1	

Table 4
State spaces of the two used agent models.

Model	Agent state includes states of	Node states	State space size
Large	All nodes	*Read* & *Searched* *Read* & *Not searched* *Not read* & *Searched* *Not read* & *Not searched*	4^N
Small	One node	Same as above	$4N$

The agent model presented in [22] assumes that the states of all nodes are being recorded, leading to an exponential state space with size proportional to 4^N, where N is the number of nodes. To address the issue of exponential state space size, we propose an alternative agent model design that results in a linear increase in the state space size with the number of nodes. The main concept is to replace the notion of the agent keeping track of all node states with the assumption that it only tracks the state of the currently associated node. One way to conceptualize this setting is to envision the agent traversing the network, visiting and keeping track of one node at a time. In this new model (labeled as "Small" in Table 3 and Table 4), the size of the state space is essentially $4N$, enabling scalability for algorithms in large networks.

To associate the agent with a single node, we introduce N *Switch* actions in addition to *Read* and *Search*. These *Switch* actions correspond to the *None* actions in the "Large" model, but have a new significance in the design of the new agent model. They allow the agent to switch from the currently associated node to another node (the available switching actions depend on the graph topology). This enables the *Read* and *Search* actions to be performed on individual nodes only. The introduction of *Switch* actions is not mandatory but provides an intuitive model that reflects the agent's traversal of the graph in search of the flag.

It should be noted that the proposed "Small" model, within the adopted RL paradigm, is just one possible solution. It is motivated by the need for fast and scalable results while being limited by the original model it is based on. Its design flaws, such as the lack of contextual memory, are mitigated by the proposed communication algorithm, resulting in overall high performance. Alternative design choices, combined with the proposed algorithm (which is not limited to the "Small" model described), open up potential avenues for future research.

4.2. Adaptive consensus-based algorithm

In contrast to single agent schemes from [22], the novel distributed web hacking algorithm proposed in this research supports the dispatch of multiple agents, thus providing the potential to arrive to the desired results in a faster manner by means of collaboration. Namely, we assume a set of M autonomous agents interacting with the environment in parallel. Each of these interactions can be modeled by the same MDP described in previous subsection, related to the considered automated penetration testing task. Moreover, the agents are able to communicate information with the neighboring agents through a communication network modeled by a directed graph $\mathcal{G} = (\mathcal{M}, \mathcal{E})$, where $\mathcal{M} = \{1, \ldots, M\}$ is the set of agent nodes and $\mathcal{E} = \{(i, j)\}$ the set of directed edges (i, j). The adjacency matrix of $\mathcal{G}$ is denoted as A. The set of neighboring agents of the agent i represents the set of agents that can send information to the agent i and is denoted as $\mathcal{M}_i$; also, we shall define $\mathcal{J}_i = \mathcal{M}_i \cup \{i\}$.

The agents are aimed at solving a common task of finding the optimal action-value function in the underlying MDP (which would give them also the optimal policy for solving the set CTF problem) and to this end use the Q-learning algorithm [11]:

$$Q'_{i,t}(s,a) = Q_{i,t}(s,a) + \alpha\left(R_{i,t+1} + \gamma \max_a Q_{i,t}(S_{i,t+1}, a) - Q_{i,t}(S_{i,t}, A_{i,t})\right) \cdot \mathbf{1}_{(s,a)=(S_{i,t},A_{i,t})} \tag{3}$$

where subscript i denotes the i-th agent and subscript t denotes the time step. For simplicity, the agents are assumed to interact synchronously with the environment, but this is not necessary (e.g. the time instances in (3) can be defined as a union of all agents' time instances [29]). $Q_{i,t}(s,a)$ and $Q'_{i,t}(s,a)$ denote the action-value functions for state s and action a before and after updating,

respectively ($Q_{i,t}$ and $Q'_{i,t}$ represent the corresponding Q-matrices). $\mathbf{1}_{(s,a)=(S_{i,t},A_{i,t})}$ denotes a binary indicator function (equal to 1 if the subscript condition is satisfied and equal to 0 otherwise), so that each agent at each time step updates only action-value function connected to the currently visited state $S_{i,t}$ and action $A_{i,t}$ chosen at that state. The action is chosen according to the policy derived from $Q_{i,t}$, which is sampled based on some exploration/exploitation strategy, e.g. ϵ-greedy [11]. $R_{i,t+1}$ represents the observed reward for the action taken, α denotes the step-size parameter and γ the discount-rate parameter. It should be noted that different agents, in general, may have different exploration/exploitation strategies [13,29].

Our idea is to enable agents help each other by exchanging information on their action-value functions. Moreover, we want to incorporate some merit of "quality" of the exchanged information which can be used for appropriate weighting of the action-value functions. Inspired by the well-known eligibility traces [11], we propose that each agent keeps track of both recency and frequency of visiting the selected state/action pair within a single heuristic:

$$E'_{i,t}(s,a) = \gamma\lambda E_{i,t}(s,a) + \mathbf{1}_{(s,a)=(S_{i,t},A_{i,t})} \tag{4}$$

where $E_{i,t}(s,a)$ and $E'_{i,t}(s,a)$ denote the eligibility traces before and after updating, respectively, and λ denotes the decay-rate parameter.

Using (3) and (4), our goal is to design such a multi-step consensus communication strategy that would asymptotically, when the number of consensus steps L (performed at each time step) tends to infinity, give:

$$\lim_{L\to\infty} \bar{Q}_{i,t}^{[L]}(s,a) = \frac{\sum_{j=1}^{M} E'_{j,t}(s,a) Q'_{j,t}(s,a)}{\sum_{j=1}^{M} E'_{j,t}(s,a)} \tag{5}$$

where $\bar{Q}_{i,t}^{[L]}(s,a)$ denotes the action-value function obtained after L steps of consensus algorithm. Of course, we would like that the needed number of consensus steps L for obtaining practically effective solutions be as small as possible, and Subsection 4.3 is devoted to the solution of this problem. In view of (5), regarding the selected state/action pair, the agents that have recently and frequently updated their action-values should have relatively higher "voting" power compared to the other agents.

In order to obtain (5), we propose the following algorithm:

$$\bar{Q}_t^{[L]}(s,a) = \frac{\Gamma_t^{[L]}(s,a)}{\Sigma_t^{[L]}(s,a)} \tag{6}$$

where $\bar{Q}_t^{[L]}(s,a) = \left[\bar{Q}_{1,t}^{[L]}(s,a) \cdots \bar{Q}_{M,t}^{[L]}(s,a)\right]^T$ and the division operator is element-wise. $\Gamma_t^{[L]}(s,a)$ and $\Sigma_t^{[L]}(s,a)$ are obtained by applying L steps of consensus:

$$\begin{aligned} \Gamma_t^{[l]}(s,a) &= C\Gamma_t^{[l-1]}(s,a), \\ \Sigma_t^{[l]}(s,a) &= C\Sigma_t^{[l-1]}(s,a), \end{aligned} \tag{7}$$

$l = 1, \ldots, L$, starting from the initial conditions given by $\Gamma_t^{[0]}(s,a) = \left[E'_{1,t}(s,a)\ \ Q'_{1,t}(s,a) \cdots E'_{M,t}(s,a) Q'_{M,t}(s,a)\right]^T$ and $\Sigma_t^{[0]}(s,a) = \left[E'_{1,t}(s,a) \cdots E'_{M,t}(s,a)\right]^T$. The matrix C represents the consensus matrix, which is a row-stochastic matrix with elements such that $C(i,j) = 0$ for $j \notin \mathcal{J}_i$.

For prediction, we use the updated and consensus-based convexified action-value functions and the updated eligibility traces:

$$\begin{aligned} Q_{i,t+1}(s,a) &= \bar{Q}_{i,t}^{[L]}(s,a), \\ E_{i,t+1}(s,a) &= E'_{i,t}(s,a). \end{aligned} \tag{8}$$

One can easily obtain a more compact network-wide form of the proposed algorithm:

$$\bar{Q}_t^{[L]} = \frac{(C^L \otimes I_M) \cdot \left(E'_t \odot Q'_t\right)}{(C^L \otimes I_M) \cdot E'_t}, \tag{9}$$

where $\otimes$ denotes the Kronecker's product, I_M the identity matrix of size M, $\odot$ element-wise multiplication, $\bar{Q}_t^{[L]} = \left[(\bar{Q}_{1,t}^{[L]})^T \cdots (\bar{Q}_{M,t}^{[L]})^T\right]^T$, $E'_t = \left[E'^T_{1,t} \cdots E'^T_{M,t}\right]^T$ and $Q'_t = \left[Q'^T_{1,t} \cdots Q'^T_{M,t}\right]^T$. Since the proposed scheme in (6) and (7) is also network-wide, in order to make the presentation clearer, we give in Algorithm 1 the proposed sequence of calculations and communications from the perspective of a single agent.

4.3. Convergence speed optimization

In order for our scheme to work, it is straightforward to show that the following condition must be met:

$$\lim_{L\to\infty} C^L = \frac{\mathbf{1}\mathbf{1}^T}{M}, \tag{10}$$

Algorithm 1 Adaptive consensus-based distributed Q-learning algorithm (at time t, for agent i and state/action pair (s,a)).

Input: $Q_{i,t}(s,a)$, $E_{i,t}(s,a)$
Output: $Q_{i,t+1}(s,a)$, $E_{i,t+1}(s,a)$
Update $Q'_{i,t}(s,a)$ and $E'_{i,t}(s,a)$ using (3) and (4)
$\Gamma^{[0]}_{i,t}(s,a) = E'_{i,t}(s,a)Q'_{i,t}(s,a)$
$\Sigma^{[0]}_{i,t}(s,a) = E'_{i,t}(s,a)$
for $l = 1$ to L **do**
Send $\Gamma^{[l-1]}_{i,t}(s,a)$, $\Sigma^{[l-1]}_{i,t}(s,a)$ to all j such that $i \in \mathcal{M}_j$
Receive $\Gamma^{[l-1]}_{j,t}(s,a)$, $\Sigma^{[l-1]}_{j,t}(s,a)$ from all $j \in \mathcal{M}_i$
$\Gamma^{[l]}_{i,t}(s,a) = \sum_{j\in\mathcal{J}_i} C(i,j)\Gamma^{[l-1]}_{j,t}(s,a)$
$\Sigma^{[l]}_{i,t}(s,a) = \sum_{j\in\mathcal{J}_i} C(i,j)\Sigma^{[l-1]}_{j,t}(s,a)$
end for
$\tilde{Q}^{[L]}_{i,t}(s,a) = \Gamma^{[L]}_{i,t}(s,a)/\Sigma^{[L]}_{i,t}(s,a)$
Predict $Q_{i,t+1}(s,a)$ and $E_{i,t+1}(s,a)$ using (8)

where $\mathbf{1}$ represents a column vector of M ones. The asymptotic condition in (10) represents a classical requirement associated with consensus schemes, which is satisfied if and only if the following conditions are met:

$$\mathbf{1}^T C = \mathbf{1}^T, \;\; C\mathbf{1} = \mathbf{1}, \;\text{ and }\; \rho(C - \mathbf{1}\mathbf{1}^T/M) < 1, \tag{11}$$

where $\rho(\cdot)$ denotes the spectral radius of the matrix [30]. On top of this, we would like to make our consensus scheme as fast as possible in terms of its convergence to the asymptotic values in (5). The convergence speed is directly connected to the spectral radius; therefore, we shall design the consensus matrix C by taking into account the aforementioned requirements related to (10) and by solving the following optimization problem:

$$\text{minimize } \rho(C - \mathbf{1}\mathbf{1}^T/M) \tag{12}$$

with respect to C [30].

To this end, in order to reduce the available degrees of freedom, we adopt the assumption that C has equal non-diagonal non-zero elements in each column, meaning that whenever an agent sends some data to its neighbors the underlying consensus weights are the same, similarly as in [26] (an analogous row-wise assumption can also be considered). Formally,

$$C(i,j) = \Big(\sum_k (1 - \beta c_k A(i,k))\Big)\delta(i,j) + \beta c_j A(i,j) = I_M - \beta \tilde{L}_c, \tag{13}$$

where β is a scaling factor, c_j the elements of the normalized vector of column values $c = [\, c_1 \;\; \cdots \;\; c_M \,]^T$ ($\mathbf{1}^T c = 1$), $\delta(i,j)$ the Kronecker's delta, and $\tilde{L}_c(i,j) = \big(\sum_k c_k A(i,k)\big)\delta(i,j) - c_j A(i,j)$ is in the form of the weighted Laplacian matrix of the digraph $\mathcal{G}$. Under the assumption that $\mathcal{G}$ is strongly connected, it is straightforward to show that (10) is achieved for c obtained as a unique solution of the system of linear equations consisting of $M-1$ rows of $\tilde{L}^T c = \mathbf{0}$ and $\mathbf{1}^T c = 1$, where $\tilde{L}(i,j) = \big((\sum_k A(i,k))\delta(i,j) - A(i,j)\big)/M$ is another weighted Laplacian [26,80], e.g.

$$c = \begin{bmatrix} \tilde{L}(1,1) & \cdots & \tilde{L}(M,1) \\ \tilde{L}(1,2) & \cdots & \tilde{L}(M,2) \\ \vdots & \ddots & \vdots \\ \tilde{L}(1,M-1) & \cdots & \tilde{L}(M,M-1) \\ 1 & \cdots & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}. \tag{14}$$

Coming back to (13), we can conclude that

$$\rho(C - \mathbf{1}\mathbf{1}^T/M) = \max_{i=2,\ldots,M} |\lambda_i(C)| = \max_{i=1,\ldots,M-1} |1 - \beta\lambda_i(\tilde{L}_c)|, \tag{15}$$

where $\lambda_i(\cdot)$ denotes the eigenvalue of the matrix with the i-th largest absolute value, having in mind that the spectrum of $C - \mathbf{1}\mathbf{1}^T/M$ is the same as the spectrum of C (excluding the eigenvalue of 1 which is replaced by 0), and that from (13) it can be concluded that $\lambda_i(C) = 1 - \beta\lambda_{M-i+1}(\tilde{L}_c)$.

Now one can readily obtain, by generalizing the results from [30] to the cases of complex eigenvalues which can arise when assuming asymmetric communication topologies, that the spectral radius condition from (11) is satisfied for

$$0 < \beta < \min_i \frac{2\,\mathrm{Re}\left(\lambda_i(\tilde{L}_c)\right)}{|\lambda_i(\tilde{L}_c)|^2}, \tag{16}$$

where $\mathrm{Re}(\cdot)$ and $|\cdot|$ represent the real part and the absolute value of a complex number, respectively. Based on numerous simulations, we can state that typically the eigenvalue which minimizes the right part of (16) is $\lambda_1(\tilde{L}_c)$, in line with [30] (although this was not true in the cases of some balanced directed graph topologies). When working with symmetric graphs, (16) reduces to $0 < \beta < 2/\lambda_1(\tilde{L}_c)$ from [30]. For the optimal choice of β which achieves the fastest convergence in terms of (12), by generalizing the results from [26], we obtain:

$$\beta^* = \max\left\{ \frac{\mathrm{Re}\left(\lambda_{i^*}(\tilde{L}_c)\right)}{|\lambda_{i^*}(\tilde{L}_c)|^2}, \max_{\substack{i\neq i^* \\ |\lambda_i|\neq|\lambda_{i^*}|}} \frac{2\left(\mathrm{Re}\left(\lambda_i(\tilde{L}_c)\right) - \mathrm{Re}\left(\lambda_{i^*}(\tilde{L}_c)\right)\right)}{|\lambda_i(\tilde{L}_c)|^2 - |\lambda_{i^*}(\tilde{L}_c)|^2} \right\}, \tag{17}$$

where $i^* = \arg\min_i \mathrm{Re}(\lambda_i(\tilde{L}_c))/|\lambda_i(\tilde{L}_c)|^2$. Of course, the considered values should be within the interval defined in (16). Again, based on numerous simulations, for most cases in practice (17) reduces to $\beta^* = 2\left(\mathrm{Re}(\lambda_{M-1}(\tilde{L}_c)) - \mathrm{Re}(\lambda_1(\tilde{L}_c))\right)/\left(|\lambda_{M-1}(\tilde{L}_c)|^2 - |\lambda_1(\tilde{L}_c)|^2\right)$. In the case of symmetric graphs, this becomes $\beta^* = 2/\left(\lambda_{M-1}(\tilde{L}_c) + \lambda_1(\tilde{L}_c)\right)$ from [26,30]. Stepping back, now that we have the vector of normalized column values c (14), as well as the optimal parameter β which scales these values (17), we can readily calculate the consensus matrix C from (13) which would give the fastest convergence speed of the algorithm. One example of such a matrix will be given in the simulations section.

On a broader level, a couple of additional underlying requirements should be mentioned. For the convergence of (3), all state/action pairs should continue to be updated while the sequence of step-size parameters satisfies the usual stochastic approximation conditions [11]. Regarding the consensus scheme itself, beside the strong connectedness requirement, all neighboring agents should be able to communicate within a fixed time interval with non-zero probability [24,29,81]. This requirement allows for a wide class of communication protocols, e.g. randomized schemes, encompassing missing communications etc. These extensions of the proposed scheme are out of the scope of this paper; some of them will be assessed in the following section by means of simulations. Also, it should be noted that the underlying communication digraph $\mathcal{G}$ is fixed; possible generalizations toward time-varying communication topologies represent potential avenues for further research.

5. Simulations

In this section, we compare the characteristics of the two agent models described in Subsection 4.1 using numerical analysis. Additionally, we demonstrate the properties of the proposed distributed adaptive consensus scheme from Subsections 4.2 and 4.3, applied to the CTF problem that represents the adopted distributed web hacking task.

5.1. Agent model implementation

In the first experiment, we consider a single agent and compare the properties of the Q-learning algorithm when applied to the two agent models described below:

1. Model from [22], referred to as the “Large” state/action space model;
2. The proposed model, referred to as the “Small” state/action space model.

We conducted our simulations in *Google Colab*, building upon the *OpenAI Gym*-like web agent implementations provided in [22]. The parameter values used in the simulations were set as follows: $\alpha = 0.5$, $\gamma = 0.9$, $\epsilon = 0.5$, and $\lambda = 0.99$. The decay-rate parameter λ should be chosen based on various factors influencing the dynamics of the learning process, primarily the expected level of agreement or disagreement between different agents in the network. In general, λ should be set to a lower value as networks become smaller and the number of consensus steps increases (resulting in lower disagreement). However, through comprehensive simulations, we have observed that the proposed algorithm is highly robust to the choice of λ. Safe values typically range between 0.9 and 0.99, and for low levels of disagreement between nodes, values such as $\lambda = 0.8$ and lower can also be used.

It should be noted that when using the original model from [22], our simulations crash when the number of network nodes (web pages) exceeds 10. This crash occurs because all the available memory is exhausted while allocating space for the required Q-matrix. Although techniques like lazy loading [79] can alleviate the problem of a large state space size, it cannot be completely eliminated, as it would reappear with an increased number of nodes due to the inherent exponential complexity of the model. In the following experiments, we assume that a network consists of $N = 7$ nodes (web pages), as illustrated in Fig. 2a. Node 0 is the starting node, and node 2 contains the flag. We utilize policies obtained from the aforementioned Q-learning algorithm, applied for varying numbers of learning steps, to make predictions. The resulting episode lengths, averaged over 100 Monte-Carlo runs, are shown in Fig. 2b. As observed, the agent using the proposed small state/action space model requires significantly fewer learning iterations to achieve optimal behavior. It is important to emphasize that due to different design choices, the two mentioned models have distinct optimal policies. The corresponding action sequences are provided in Table 5. To enable a fair comparison, Fig. 2b illustrates the relative episode lengths (with respect to the corresponding optimal policy’s episode lengths).

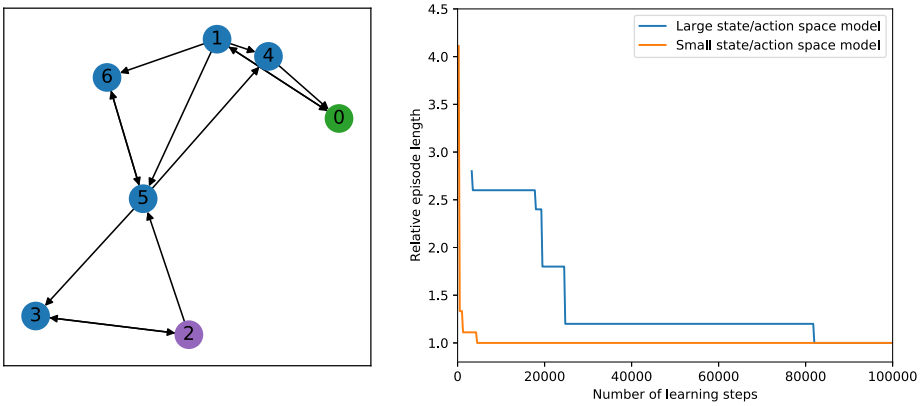
5.2. Consensus-based distributed algorithm

In the second experiment, we utilize the adopted agent model within a distributed multi-agent scenario, considering a larger network. Specifically, we construct a network consisting of $N = 50$ web pages organized into $M = 5$ websites, with 10 web pages in each website. Agents are associated with websites, such that an agent’s starting state is connected to a web page within its corresponding website. We establish connections between the agents by considering their inter-distances and randomly terminate each link with a 50% probability, ensuring the preservation of the strong connectedness requirement. This process yields the directed communication topology for the agents.

Both the node network and the agent network are illustrated in Fig. 3a. It is important to note that the proposed consensus algorithm is not limited to the specific setting of agents connected to websites. The chosen example serves for illustrative purposes,

Table 5
Action sequences for the optimal policies obtained for the two agent models used and the network with 7 nodes (web pages) illustrated in Fig. 2a.

Model	Large	Small
Optimal policy action sequence	*Read (node 0 = start node)* *Read (node 1)* *Read (node 5)* *Read (node 3)* *Search (node 2 = flag node)*	*Read (current node = start node 0)* *Switch (to node 1)* *Read (current node)* *Switch (to node 5)* *Read (current node)* *Switch (to node 3)* *Read (current node)* *Switch (to node 2)* *Search (current node = flag node 2)*



(a) Graph topology (b) Average episode lengths

Fig. 2. Single agent traversing 7 nodes using different state space models. (left) The network representation of a website, with nodes (web pages) depicted as blue circles. The starting node is highlighted in green, and the node containing the flag is shown in purple. The node links are indicated by directed arrows. (right) Corresponding average episode lengths, relative to the episode lengths of corresponding optimal policies, for policies obtained using the Q-learning algorithm with varying numbers of steps. The results are shown for two different agent models. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

inspired by potential analogies in practical scenarios. The only limitation is that the agents interact with the environment through copies of the same MDP model.

We determine the optimal consensus parameters in terms of convergence speed by leveraging equations (13), (14) and (17). Specifically, for the given network topology, we determine that $c = [1/5, 1/15, 2/15, 1/3, 4/15]^T$ and $\beta^* = 2.0582$. Therefore, the consensus matrix to be used is as follows:

$$C = \begin{bmatrix} 0.5884 & 0.1372 & 0.2744 & 0 & 0 \\ 0.4116 & 0.5884 & 0 & 0 & 0 \\ 0 & 0 & 0.4512 & 0 & 0.5488 \\ 0 & 0.1372 & 0 & 0.3139 & 0.5488 \\ 0 & 0.1372 & 0.2744 & 0.6861 & -0.0977 \end{bmatrix}.$$

The consensus matrix above is asymmetric due to the assumed asymmetric adjacency matrix, corresponding to the network topology depicted in the top left part of Fig. 3a. Furthermore, there is a negative communication weight, which aligns with the reasoning presented in [30].

We compare properties of the Q-learning algorithm within four different schemes:

1. Non-cooperative parallel multi-agent scheme: In this scheme, agents work individually without any communication, representing M independent single-agent schemes (referred to as “No consensus”).
2. Cooperative distributed multi-agent scheme with average consensus: This scheme assumes communication between neighboring agents and implements the average consensus algorithm with $L = 5$ consensus steps (referred to as “Average consensus”).
3. Cooperative distributed multi-agent scheme with adaptive consensus: Similar to scheme 2, but here the proposed adaptive consensus scheme is used instead of the average consensus algorithm (referred to as “Adaptive consensus”).
4. Centralized scheme: This scheme represents a single entity that collects all information from all agents and updates the Q-table accordingly (referred to as “Centralized”).

For scheme 3, we initialize all values in the eligibility traces matrices to 0.01. Furthermore, we enforce that these values cannot fall below 0.01, which is a convenient constraint to prevent numerical underflows and overflows when calculating equation (6).

In addition, we utilize the existing implementation [31] of the two algorithms assuming function approximation (FA):

5. Synchronous, deterministic variant of the Asynchronous Advantage Actor Critic algorithm (referred to as "FA-A2C").
6. Proximal Policy Optimization algorithm (referred to as "FA-PPO").

Both of these FA algorithms use multiple workers ($M = 5$). Their learning rate and the number of learning steps per update are tuned to optimize performance. FA algorithms with Graph Neural Networks (GNNs) are an attractive alternative [82] due to their ability to capture local and global graph patterns. While GNNs have been utilized in the considered application domain [83], their integration within the RL formalism remains an interesting topic for future research. Meanwhile, the performance of FA-A2C and FA-PPO may be indicative of the performance of corresponding GNN-based algorithms, given that they all employ deep neural network approximators.

The agent policies obtained by the listed algorithms are used for prediction. Fig. 3b displays the resulting average episode lengths per agent, additionally averaged over 100 Monte-Carlo runs, plotted against the number of learning steps per agent used for obtaining the policies. The non-cooperative scheme (without consensus) predictably exhibits the poorest performance among all the Q-learning approaches. Fig. 3b shows that the proposed adaptive consensus scheme significantly accelerates the learning process compared to the average consensus algorithm, as it reduces the number of learning steps required to achieve a near-optimal solution by a factor of 2 to 3. Its performance is very close to the performance of the centralized scheme. The two FA algorithms (A2C and PPO) require more learning steps to approach optimal solutions, likely due to their gradient descent optimization dynamics, which necessitates too many samples for successful training, making them less sample efficient.

5.3. Exploring algorithm performance and properties

To evaluate the performance of the proposed algorithm across different network topologies, we include several networks with $N = 50$ nodes in our simulation and employ $M = 5$ agents to explore these networks. We specifically examine the following topologies:

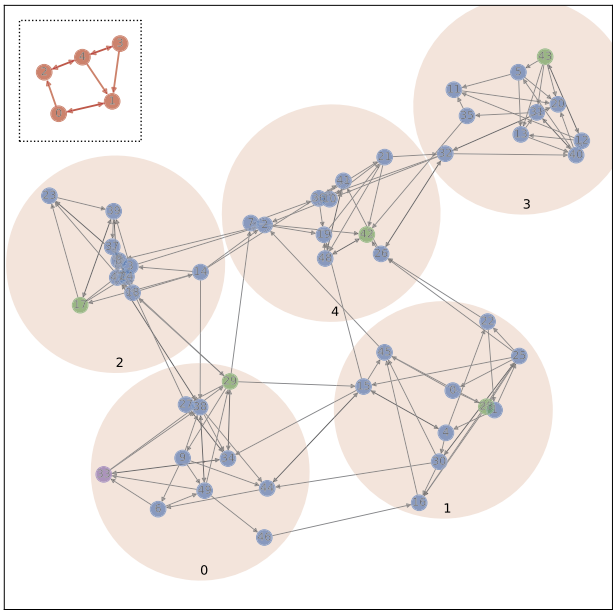
1. Erdős–Rényi graph [84]: This graph is designed by randomly connecting nodes such that the number of links is approximately equal to the number of links in Fig. 3a while ensuring strong connectedness (see Fig. 4a).
2. Scale-free graph [85]: This graph represents a strongly connected component of a larger initial scale-free graph, with some nodes having a large number of connections (hubs) and the majority of nodes having a small number of connections (see Fig. 4c).
3. Wiskonsin web graph: This graph is obtained as a strongly connected component of an actual web graph [86] (see Fig. 4e).

Nodes are associated with corresponding agents through a suitable clustering process. The relative performance of the observed algorithms applied to the Erdős–Rényi (Fig. 4b) and the Wisconsin (Fig. 4f) web graphs remains consistent with that shown in Fig. 3b. Interestingly, in the case of the Scale-free graph (Fig. 4d), the performance of the FA algorithms is relatively similar to that of the proposed and centralized algorithms, albeit slower in terms of convergence toward optimal solutions. Moreover, it can be seen that the introduction of the proposed adaptive consensus scheme significantly enhances the performance of the learning algorithm. Without adaptation or with the average consensus scheme, the corresponding performance falls short of approaching optimal solutions throughout the experiment.

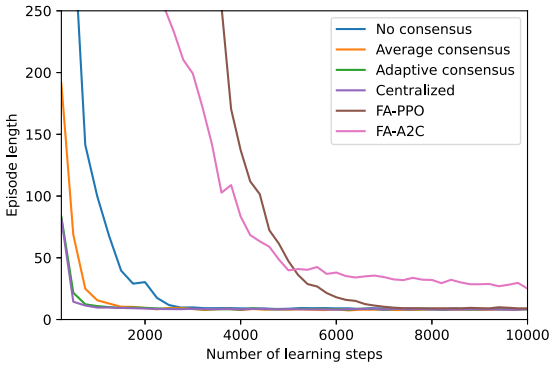
For a deeper analysis of the performance of different communication schemes, we implement several additional communication protocols:

1. Randomized single broadcast algorithm: At a given communication event, a single node i asynchronously broadcasts information to a randomly selected neighbor j such that $i \in \mathcal{M}_j$ [87] (referred to as "Broadcast single").
2. Randomized multi-broadcast algorithm: At a given communication event, a single node i asynchronously broadcasts information to all of its neighbors j such that $i \in \mathcal{M}_j$ [88] (referred to as "Broadcast multi").
3. The "communicating transitions" algorithm: This algorithm synchronously exchanges raw data, namely, actual transitions $(S_{i,t}, A_{i,t}, R_{i,t+1}, S_{i,t+1})$ between all nodes in a neighbor-wise fashion (referred to as "Communicating transitions").

The choice of the above broadcast communication protocols 1 and 2 is motivated by the adopted asymmetric agent network topology. These algorithms employ the so-called companion or surplus variables necessary for the consensus protocol to converge for each sample path of the communication sequence [87,88]. For the experimental design, we employ a larger setting, similar to Fig. 3a, but with $N = 250$ nodes and $M = 25$ agents, to clearly differentiate the performance of the resulting algorithms. Fig. 5 shows the results. The FA algorithms did not yield comparable performance. The consensus algorithms from Subsection 5.2 are labeled "synchronous" as they assume that, at a given time, all nodes exchange information with all their neighbors, as specified by Algorithm 1. The proposed synchronous algorithm (green line) performs better than the corresponding randomized schemes (green dashed and dotted lines), which, in turn, outperform all average consensus schemes (yellow lines). Therefore, using asynchronous variants of the proposed algorithm also yields benefits over average consensus algorithms. The "Communicating transitions" algorithm (red line) outperforms the average consensus schemes but falls short of the adaptive consensus schemes in performance.



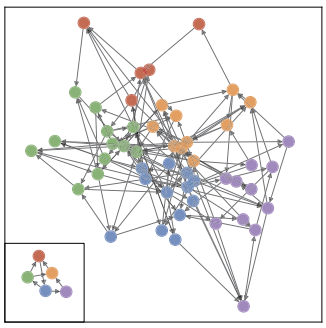
(a) Graph topology



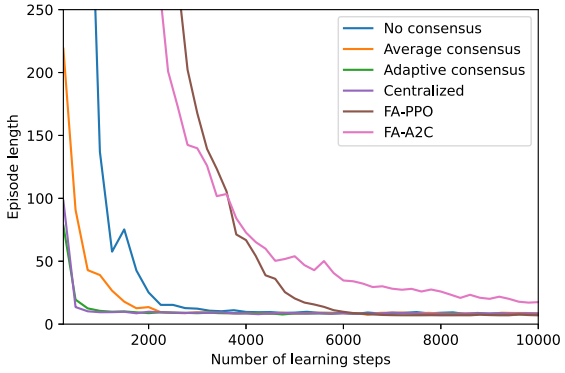
(b) Corresponding average episode lengths

Fig. 3. Traversal of 50 nodes by five agents using different algorithms. (top) The graphs provide an overview of a network structure and connections between websites, agents, and web pages, enabling a comprehensive understanding of the traversal and information flow in the system. The network represents a collection of websites (large red-filled circles) with nodes, i.e., web pages, appearing as circles colored according to their association with different agent positions. Each web page (node) is associated with its respective website hosted on a web server. The starting nodes of the agents are colored green, and the node carrying the flag is purple. The remaining nodes, representing the web pages to be traversed by the agents, are depicted as blue circles. The links between nodes are represented by directed black arrows. Connected to each website is an agent, symbolized by a red circle in the agent network shown in the top-left corner. Agent links are depicted by directed red arrows, denoting the relationships and communication channels between agents. (bottom) The diagram provides insights into the performance and convergence rates of the tested algorithms by displaying episode lengths as the Q-learning process progresses. It compares the average episode lengths (y-axis) of policies to the number of steps (x-axis) taken by the Q-learning algorithm to derive these policies, for different algorithms used in the study.

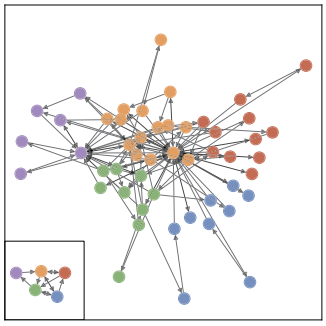
Finally, to gain insight into how the proposed adaptive consensus scheme operates, in the case of learning task from Fig. 3a, we select one state from the corresponding Q-tables (i.e. node 49 *Read* & *Searched* belonging to website 0), and plot the evolution of all the corresponding action values that are updated for each agent (Fig. 6). From left to right, the plots in Fig. 6 depict the action value evolution for no consensus, average consensus, and adaptive consensus algorithms, respectively. Notice that in each plot, every five lines of the same color represent the estimated expected returns for the same action assigned to five communicating agents. These lines mostly overlap in the average consensus (center) and adaptive consensus (right) plots, even after a relatively small number of consensus steps are applied at each learning step. However, in the no consensus plot (left), the lines are significantly spaced from each other. Their spacing reflects agents' initial distances from the flag (set here at node 33). On average, agents starting closer to the flag exhibit earlier increases in their action values. The proposed adaptive consensus scheme produces lines that are close to the lines of the "fastest" agents in the non-cooperative case. On the other hand, the average consensus scheme, as the name suggests, produces lines that are close to the averages of the corresponding lines in the non-cooperative case. Notably, actions directed towards the flag carrying node 33 have the highest values. It should be noted that in the case of larger agent networks and sparser communication topologies, a larger number of consensus steps L should be used. However, due to the specific properties of the used Q-learning



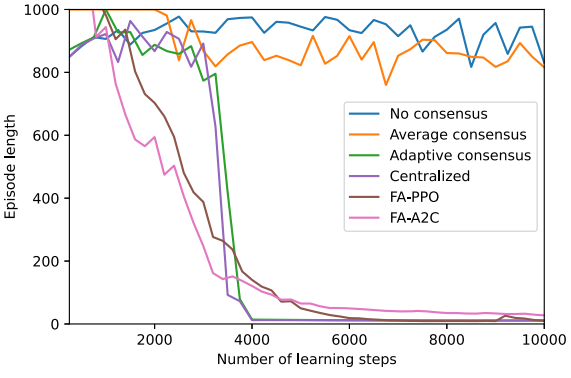
(a) Erdős–Rényi graph



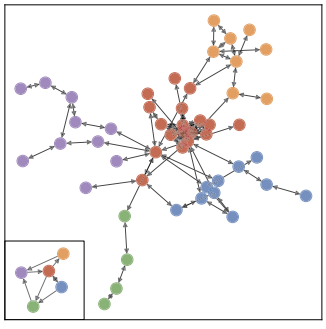
(b) Corresponding average episode lengths



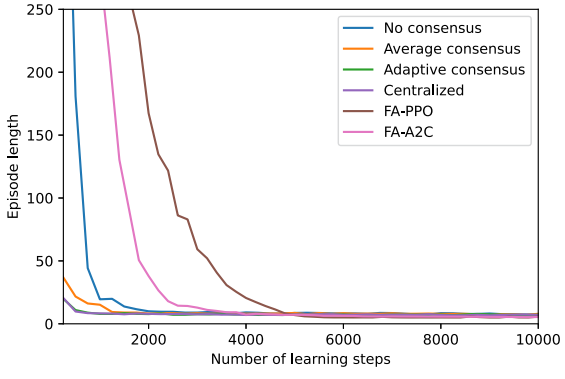
(c) Scale-free graph



(d) Corresponding average episode lengths



(e) Wiskonsin web graph



(f) Corresponding average episode lengths

Fig. 4. Five agents traverse 50-node networks with different topologies. (left) Networks where nodes are colored according to their association with separate agents, with the agent network also depicted at the bottom-left. (right) The average episode lengths of policies are shown in relation to the number of steps taken by the Q-learning algorithm to obtain these policies, for different algorithms used in the study.

algorithm, particularly its inherently sparse updates, even in cases of very large agent networks, the required number of consensus steps to achieve a sufficient level of agreement between agents should not be excessively high.

6. Conclusion

In this paper, we introduce a novel consensus-based learning algorithm designed for distributed web hacking. The algorithm employs an adaptive communication scheme that ensures all agents in the distributed multi-agent reinforcement learning context receive intended network-wide information in the fastest decentralized manner, as proven by our research.

We conducted a detailed analysis of the convergence properties of the proposed algorithm, specifically focusing on the general case of asymmetric communication topologies. Additionally, we proposed a web agent model for automated penetration testing,

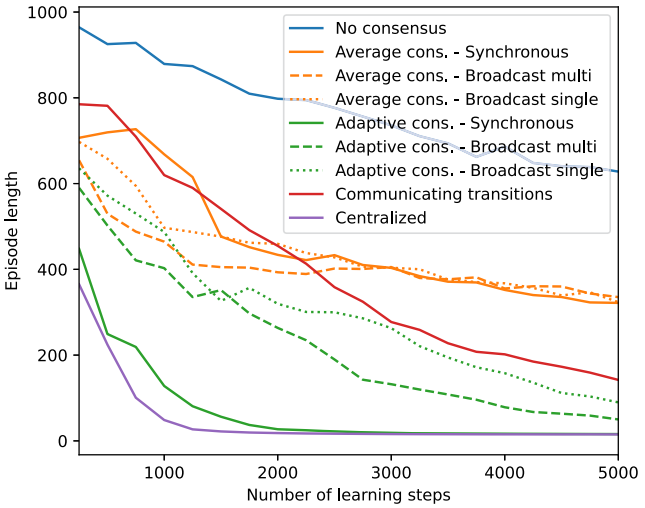


Fig. 5. Twenty-five agents traversing a 250-node network, employing different communication schemes. The average episode lengths of policies are plotted against the number of steps taken by the Q-learning algorithm to obtain these policies, across different algorithms.

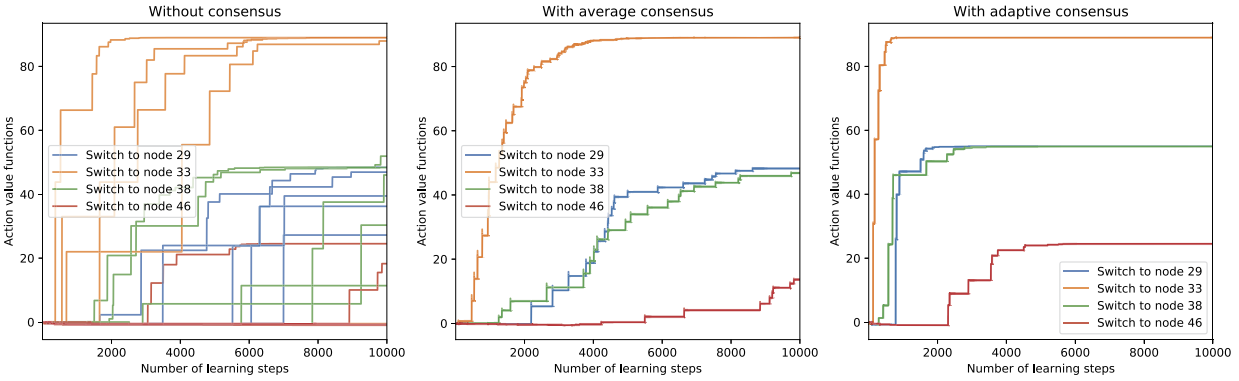


Fig. 6. Time evolution of the action value functions that receive updates for one selected state (i.e., node 49 *Read* & *Searched*) connected to different agents (different lines of the same color) for different algorithms.

which exhibits enhanced scalability properties. We also presented a comprehensive systematic review of the existing literature in this area.

To validate our approach, we performed numerical simulations considering various network topologies and communication protocols. The results indicate that our algorithm outperforms analogous average consensus algorithms and more complex baseline algorithms, significantly reducing hacking times. The obtained results can be used by system security practitioners, enabling them to enhance web security by rapidly and early detecting vulnerabilities.

The work presented in this paper opens up several possible directions for further research. The proposed consensus scheme can operate with more complex agent models (e.g., Levels 2 - 7 from [22]). Function approximation models can tackle large state-action spaces, similar to the approach taken in [24]. In this context, the agents would exchange information on the parameters of the models approximating either the action value or policy functions, or both. The main challenge would be determining the relevant quantities for weighting the adaptive communication scheme.

Distributing the optimization process or designing optimized consensus algorithms under different assumptions, such as dynamically changing interaction topologies or randomized communication, should also be explored in the future. Finally, comparing the proposed solution with the alternatives based on Graph Neural Networks (GNNs) would provide insights of practical importance for future design.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This paper is the result of research performed by Artificial Intelligence Department within Vlatacom Institute of High Technologies, Belgrade, Serbia. Aspects of the research described in this article are the subject of the pending patent application RS/P-2023-0963 by the same authors.

References

[1] Cost of a data breach, Tech. Rep., IBM Corporation, 2020.
[2] S. Morgan, Cybercrime damages $6 trillion by 2021, https://cybersecurityventures.com/annual-cybercrime-report-2017/, October 2017. (Accessed 31 January 2022).
[3] A. O'Neill, Growth of the global gross domestic product (GDP) 2026, https://www.statista.com/statistics/268750/global-gross-domestic-product-gdp/, January 2022. (Accessed 31 January 2022).
[4] Verizon, Data breach investigations report, Tech. Rep., Verizon, 2020.
[5] S. Shackelford, From nuclear war to net war: analogizing cyber attacks in international law, Berkley J. Int. Law 25 (3) (2009).
[6] D.E. Denning, Stuxnet: what has changed?, Future Internet 4 (3) (2012) 672–687, https://doi.org/10.3390/fi4030672, https://www.mdpi.com/1999-5903/4/3/672.
[7] S. Sanders, Massive data breach puts 4 million federal employees' records at risk, https://www.npr.org/sections/thetwo-way/2015/06/04/412086068/massive-data-breach-puts-4-million-federal-employees-records-at-isk, June 2015. (Accessed 31 January 2022).
[8] L.H. Newman, Hacking diplomatic cables is expected. Exposing them is not, https://www.wired.com/story/eu-diplomatic-cable-hacks-area-one/, December 2018. (Accessed 31 January 2022).
[9] Appsec stats flash, Tech. Rep. 2, WhiteHat Security, Inc, February 2021.
[10] J. Johnson, Human-initiated and automated bot attacks volume worldwide 2020, by region, https://www.statista.com/statistics/1180124/human-initiated-automated-bot-attacks-volume-worldwide-region/, January 2021. (Accessed 31 January 2022).
[11] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 2nd edition, The MIT Press, Cambridge, Massachusetts, 2018.
[12] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A.D. Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu, D. Silver, Massively parallel methods for deep reinforcement learning, arXiv:1507.04296, 2015.
[13] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: M.F. Balcan, K.Q. Weinberger (Eds.), Proceedings of the 33rd International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 48, PMLR, New York, New York, USA, 2016, pp. 1928–1937, https://proceedings.mlr.press/v48/mniha16.html.
[14] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, K. Kavukcuoglu, IMPALA: scalable distributed deep-RL with importance weighted actor-learner architectures, arXiv:1802.01561, 2018.
[15] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, D. Silver, Distributed prioritized experience replay, arXiv:1803.00933, 2018.
[16] J. Foerster, I.A. Assael, N. de Freitas, S. Whiteson, Learning to communicate with deep multi-agent reinforcement learning, in: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 29, Curran Associates, Inc., 2016, https://proceedings.neurips.cc/paper_files/paper/2016/file/c7635bfd99248a2cdef8249ef7bfbef4-Paper.pdf.
[17] T. Eccles, Y. Bachrach, G. Lever, A. Lazaridou, T. Graepel, Biases for emergent communication in multi-agent reinforcement learning, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper_files/paper/2019/file/fe5e7cb609bdbe6d62449d61849c38b0-Paper.pdf.
[18] L. Matignon, G. Laurent, N. Fort-Piat, Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems, Knowl. Eng. Rev. 27 (2012) 1–31, https://doi.org/10.1017/S0269888912000057.
[19] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017, https://proceedings.neurips.cc/paper_files/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.
[20] R. Olfati-Saber, A. Fax, R. Murray, Consensus and cooperation in networked multi-agent systems, Proc. IEEE 95 (2007) 215–233.
[21] W. Ren, R.W. Beard, E.M. Atkins, A survey of consensus problems in multi-agent coordination, in: Proc. American Control Conference, 2005, pp. 1859–1864.
[22] L. Erdődi, F.M. Zennaro, The agent web model: modeling web hacking for reinforcement learning, Int. J. Inf. Secur. 06 (2021), https://doi.org/10.1007/s10207-021-00554-7.
[23] S. Kar, J.M. Moura, H.V. Poor, *QD*-learning: a collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations, IEEE Trans. Signal Process. 61 (7) (2013) 1848–1862, https://doi.org/10.1109/TSP.2013.2241057.
[24] M.S. Stanković, M. Beko, S.S. Stanković, Distributed value function approximation for collaborative multi-agent reinforcement learning, IEEE Trans. Control Netw. Syst. (2021).
[25] N. Ilić, M.S. Stanković, S.S. Stanković, Adaptive consensus-based distributed target tracking in sensor networks with limited sensing range, IEEE Trans. Control Syst. Technol. 22 (2) (2014) 778–785, https://doi.org/10.1109/TCST.2013.2256787.
[26] S. Stanković, N. Ilić, M.S. Stanković, Adaptive consensus-based distributed system for multisensor multitarget tracking, IEEE Trans. Aerosp. Electron. Syst. (2021) 1, https://doi.org/10.1109/TAES.2021.3132285.
[27] S. Valcarcel Macua, A. Tukiainen, D. García-Ocaña Hernández, D. Baldazo, E. Munoz de Cote, S. Zazo, Diff-DAC: distributed actor-critic for average multitask deep reinforcement learning, in: Adaptive Learning Agents workshop (ALA2018), 2018, https://doi.org/10.48550/arxiv.1710.10363.
[28] W. Suttle, Z. Yang, K. Zhang, Z. Wang, T. Başar, J. Liu, A multi-agent off-policy actor-critic algorithm for distributed reinforcement learning, in: 21st IFAC World Congress, IFAC-PapersOnLine 53 (2) (2020) 1549–1554, https://doi.org/10.1016/j.ifacol.2020.12.2021.
[29] D. Dašić, N. Ilić, M. Vučetić, M. Perić, M. Beko, M.S. Stanković, Distributed spectrum management in cognitive radio networks by consensus-based reinforcement learning, Sensors 21 (9) (2021), https://doi.org/10.3390/s21092970, https://www.mdpi.com/1424-8220/21/9/2970.
[30] L. Xiao, S. Boyd, Fast linear iterations for distributed averaging, Syst. Control Lett. 53 (2004) 65–78.
[31] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Stable-Baselines3: reliable reinforcement learning implementations, J. Mach. Learn. Res. 22 (268) (2021) 1–8, http://jmlr.org/papers/v22/20-1364.html.
[32] G.I. Simari, From data to knowledge engineering for cybersecurity, in: IJCAI International Joint Conference on Artificial Intelligence 2019-August, 2019, pp. 6403–6407.

[33] S.V. Albrecht, P. Stone, Autonomous agents modelling other agents: a comprehensive survey and open problems, Artif. Intell. 258 (2018) 66–95, https://doi.org/10.1016/j.artint.2018.01.002, https://www.sciencedirect.com/science/article/pii/S0004370218300249.
[34] K. Pozdniakov, E. Alonso, V. Stankovic, K. Tam, K. Jones, Smart security audit: reinforcement learning with a deep neural network approximator, in: 2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), 2020, pp. 1–8.
[35] G.F. Lyon, Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning, 2009, Insecure, Sunnyvale, CA, USA.
[36] R. Sharpe, E. Warnicke, U. Lamping, Wireshark user's guide, https://www.wireshark.org/docs/wsug_html_chunked. (Accessed 31 January 2022).
[37] John the Ripper password cracker, https://www.openwall.com/john. (Accessed 31 January 2022).
[38] Burp suite, https://portswigger.net/burp. (Accessed 31 January 2022).
[39] Rapid7, Metasploit - the world's most used penetration testing framework, https://www.metasploit.com. (Accessed 31 January 2022).
[40] Offensive Security, Kali Linux, https://www.kali.org. (Accessed 31 January 2022).
[41] M. Boddy, J. Gohde, T. Haigh, S. Harp, Course of action generation for cyber security using classical planning, in: Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling, ICAPS'05, AAAI Press, 2005, pp. 12–21.
[42] J.L. Obes, C. Sarraute, G. Richarte, Attack planning in the real world, ArXiv, arXiv:1306.4044 [abs], 2013.
[43] C. Sarraute, G. Richarte, J. Lucángeli Obes, An algorithm to find optimal attack paths in nondeterministic scenarios, in: Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence, AISec '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 71–80, https://doi.org/10.1145/2046684.2046695.
[44] C. Phillips, L.P. Swiler, A graph-based system for network-vulnerability analysis, in: Proceedings of the 1998 Workshop on New Security Paradigms, NSPW '98, Association for Computing Machinery, New York, NY, USA, 1998, pp. 71–79, https://doi.org/10.1145/310889.310919.
[45] C. Sarraute, O. Buffet, J. Hoffmann, Penetration testing = = pomdp solving? ArXiv, arXiv:1306.4714 [abs], 2013.
[46] C. Sarraute, O. Buffet, J. Hoffmann, POMDPs make better hackers: accounting for uncertainty in penetration testing, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI'12, AAAI Press, 2012, pp. 1816–1824.
[47] J. Schwartz, H. Kurniawati, E. El-Mahassni, POMDP + information-decay: incorporating defender's behaviour in autonomous penetration testing, in: Proceedings of the International Conference on Automated Planning and Scheduling, vol. 30(1), 2020, pp. 235–243, https://ojs.aaai.org/index.php/ICAPS/article/view/6666.
[48] R. Böhme, M. Félegyházi, Optimal information security investment with penetration testing, in: T. Alpcan, L. Buttyán, J.S. Baras (Eds.), Decision and Game Theory for Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 21–37.
[49] J. Robertson, A. Diab, E. Marin, E. Nunes, V. Paliath, J. Shakarian, P. Shakarian, Using Game Theory for Threat Intelligence, Cambridge University Press, 2017, pp. 67–95.
[50] P. Speicher, M. Steinmetz, J. Hoffmann, M. Backes, R. Künnemann, Towards automated network mitigation analysis, in: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1971–1978, https://doi.org/10.1145/3297280.3297473.
[51] E.J. Colbert, A. Kott, L.P. Knachel, The game-theoretic model and experimental investigation of cyber wargaming, J. Defense Model. Simul. 17 (1) (2020) 21–38, https://doi.org/10.1177/1548512918795061, https://doi.org/10.1177/1548512918795061.
[52] A. Nisioti, G. Loukas, S. Rass, E. Panaousis, Game-theoretic decision support for cyber forensic investigations, Sensors 21 (16) (2021), https://doi.org/10.3390/s21165300, https://www.mdpi.com/1424-8220/21/16/5300.
[53] M.C. Ghanem, T.M. Chen, Reinforcement learning for intelligent penetration testing, in: 2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2018, pp. 185–192.
[54] M.C. Ghanem, T.M. Chen, Reinforcement learning for efficient network penetration testing, Information 11 (1) (2020), https://doi.org/10.3390/info11010006, https://www.mdpi.com/2078-2489/11/1/6.
[55] M.C. Ghanem, T.M. Chen, E.G. Nepomuceno, Hierarchical reinforcement learning for efficient and effective automated penetration testing of large networks, J. Intell. Inf. Syst. 60 (2) (2023) 281–303.
[56] J. Schwartz, H. Kurniawati, Autonomous penetration testing using reinforcement learning, https://doi.org/10.48550/arXiv.1905.05965, 2019.
[57] T.-y. Zhou, Y.-c. Zang, J.-h. Zhu, Q.-x. Wang, NIG-AP: a new method for automated penetration testing, Front. Inf. Technol. Electron. Eng. 20 (9) (2019) 1277–1288, https://doi.org/10.1631/FITEE.1800532.
[58] S. Zhou, J. Liu, D. Hou, X. Zhong, Y. Zhang, Autonomous penetration testing based on improved deep Q-network, Appl. Sci. 11 (19) (October 2021), https://doi.org/10.3390/app11198823.
[59] Z. Hu, R. Beuran, Y. Tan, Automated penetration testing using deep reinforcement learning, in: 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW), 2020, pp. 2–10.
[60] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, J. Kautz, Reinforcement learning through asynchronous advantage actor-critic on a GPU, in: Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 2017.
[61] H. Nguyen, S. Teerakanok, A. Inomata, T. Uehara, The proposal of double agent architecture using actor-critic algorithm for penetration testing, in: Proceedings of the 7th International Conference on Information Systems Security and Privacy, 2021, pp. 440–449.
[62] S. Chakraborty, R. Krishna, Y. Ding, B. Ray, Deep learning based vulnerability detection: are we there yet?, IEEE Trans. Softw. Eng. 48 (09) (2022) 3280–3296, https://doi.org/10.1109/TSE.2021.3087402.
[63] C. Chebbi, Mastering Machine Learning for Penetration Testing, Packt Publishing, 2018.
[64] D. Son, Deep exploit: fully automatic penetration test tool using machine learning, https://securityonline.info/deep-exploit/, May 2020. (Accessed 31 January 2022).
[65] D.R. McKinnel, T. Dargahi, A. Dehghantanha, K.-K.R. Choo, A systematic literature review and meta-analysis on artificial intelligence in penetration testing and vulnerability assessment, Comput. Electr. Eng. 75 (2019) 175–188, https://doi.org/10.1016/j.compeleceng.2019.02.022, https://www.sciencedirect.com/science/article/pii/S0045790618315489.
[66] C. Sarraute, O. Buffet, J. Hoffmann, POMDPs make better hackers: accounting for uncertainty in penetration testing, in: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, AAAI'12, AAAI Press, 2012, pp. 1816–1824.
[67] J. Andress, The Basics of Information Security: Understanding the Fundamentals of InfoSec in Theory and Practice, 2nd edition, Elsevier Inc., 2014.
[68] Y. Liu, Y. Morgan, Security against passive attacks on network coding system – a survey, Comput. Netw. 138 (2018) 57–76, https://doi.org/10.1016/j.comnet.2018.03.013, https://www.sciencedirect.com/science/article/pii/S138912861830121X.
[69] M. Babiker, E. Karaarslan, Y. Hoscan, Web application attack detection and forensics: a survey, in: 2018 6th International Symposium on Digital Forensic and Security (ISDFS), 2018, pp. 1–6.
[70] O.B. Fredj, O. Cheikhrouhou, M. Krichen, H. Hamam, A. Derhab, An OWASP top ten driven survey on web application protection methods, in: Risks and Security of Internet and Systems, Springer International Publishing, 2021.
[71] M.N. Khalid, H. Farooq, M. Iqbal, M.T. Alam, K. Rasheed, Predicting web vulnerabilities in web applications based on machine learning, in: I.S. Bajwa, F. Kamareddine, A. Costa (Eds.), Intelligent Technologies and Applications, Springer Singapore, Singapore, 2019, pp. 473–484.
[72] S.K. Lala, A. Kumar, S. T., Secure web development using OWASP guidelines, in: 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 323–332.
[73] A. Singh, A. Sharma, N. Sharma, I. Kaushik, B. Bhushan, Taxonomy of attacks on web based applications, in: 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), vol. 1, 2019, pp. 1231–1235.

[74] J. Li, Vulnerabilities mapping based on OWASP-SANS: a survey for static application security testing (SAST), Ann. Emerg. Technol. Comput. 4 (3) (2020) 1–8, https://doi.org/10.33166/aetic.2020.03.001.
[75] N. Bhateja, S. Sikka, A. Malhotra, A Review of SQL Injection Attack and Various Detection Approaches, John Wiley & Sons, Ltd, 2021, pp. 481–489, https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119752134.ch34, https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119752134.ch34.
[76] Open Web Application Security Project (OWASP), OWASP top 10 - 2021 the ten most critical web application security risks, https://owasp.org/www-project-top-ten/, 2021.
[77] Y. Sadqi, Y. Maleh, A systematic review and taxonomy of web applications threats, Inf. Secur. J., Glob. Perspect. 31 (1) (2022) 1–27, https://doi.org/10.1080/19393555.2020.1853855.
[78] D. Fraze, Cyber grand challenge, https://www.darpa.mil/program/cyber-grand-challenge, 2016. (Accessed 31 January 2022).
[79] F.M. Zennaro, L. Erdődi, Modelling penetration testing with reinforcement learning using capture-the-flag challenges: trade-offs between model-free learning and a priori knowledge, IET Inf. Secur. 17 (3) (2023) 441–457, https://doi.org/10.1049/ise2.12107, https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/ise2.12107, https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/ise2.12107.
[80] M.S. Stanković, N. Ilić, S.S. Stanković, Distributed stochastic approximation: weak convergence and network design, IEEE Trans. Autom. Control 61 (12) (2016) 4069–4074.
[81] M.S. Stanković, M. Beko, S.S. Stanković, Distributed gradient temporal difference off-policy learning with eligibility traces: weak convergence, in: Proc. IFAC World Congress, 2020.
[82] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, A. Cabellos-Aparicio, Deep reinforcement learning meets graph neural networks: exploring a routing optimization use case, Comput. Commun. 196 (2022) 184–194, https://doi.org/10.1016/j.comcom.2022.09.029, https://www.sciencedirect.com/science/article/pii/S0140366422003784.
[83] W.W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, M. Portmann, E-GraphSAGE: a graph neural network based intrusion detection system for IoT, in: NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2022, https://doi.org/10.1109/noms54207.2022.9789878.
[84] P. Erdös, A. Rényi, On random graphs I, Publ. Math. (Debr.) 6 (1959) 290.
[85] B. Bollobás, C. Borgs, J.T. Chayes, O. Riordan, Directed scale-free graphs, in: ACM-SIAM Symposium on Discrete Algorithms, 2003.
[86] T. de Paula Peixoto, WebKB graphs, https://networks.skewed.de/net/webkb, 1998. (Accessed 29 January 2023).
[87] K. Cai, H. Ishii, Average consensus on general strongly connected digraphs, Automatica 48 (11) (2012) 2750–2761, https://doi.org/10.1016/j.automatica.2012.08.003, https://www.sciencedirect.com/science/article/pii/S0005109812004049.
[88] S. Wu, M.G. Rabbat, Broadcast gossip algorithms for consensus on strongly connected digraphs, IEEE Trans. Signal Process. 61 (2012) 3959–3971.