International Journal of Electrical and Computer Engineering (IJECE)
Vol. 14, No. 1, February 2024, pp. 772~781
ISSN: 2088-8708, DOI: 10.11591/ijece.v14i1.pp772-781

Survey on detecting and preventing web application broken access control attacks

Ahmed Anas, Salwa Elgamal, Basheer Youssef
Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt

Article Info

Article history:

Received Jul 27, 2023
Revised Oct 3, 2023
Accepted Oct 9, 2023

Keywords:

Broken access control
Exploitation detection
Logical vulnerabilities
Survey paper
Web application security

ABSTRACT

Web applications are an essential component of the current wide range of digital services proposition including financial and governmental services as well as social networking and communications. Broken access control vulnerabilities pose a huge risk to that echo system because they allow the attacker to circumvent the allocated permissions and rights and perform actions that he is not authorized to perform. This paper gives a broad survey of the current research progress on approaches used to detect access control vulnerabilities exploitations and attacks in web application components. It categorizes these approaches based on their key techniques and compares the different detection methods in addition to evaluating their strengths and weaknesses. We also spotted and elaborated on some exciting research gaps found in the current literature, Finally, the paper summarizes the general detection approaches and suggests potential research directions for the future.

Corresponding Author:

Ahmed Anas
Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University
5 Dr. Ahmed Zewail Street, Orman, Giza, Egypt
Email: ahmed.anas@pg.fci-cu.edu.eg

1. INTRODUCTION

Web applications are software programs that run on a web server and can be accessed over the internet through a variety of clients such as browsers or mobile apps. They are essential nowadays and have emerged as a primary service delivery channel. Customers and residents can use them as a platform to access various utilities, including banking, shopping, governmental services, social networking, and professional and personal communication.

Web applications and web application programming interfaces (APIs) are suspectable to various types of vulnerabilities due to their inherent structure as it is reachable from external sources and networks. Web applications are attractive targets for attackers as they may be used to run critical business systems and contain sensitive data. Accordingly, it is important to have robust countermeasures in place to detect and prevent exploitation. Malicious actors often target web applications exploiting the possible various vulnerabilities due to the lack of adherence to secure coding and architecture best practices.

Generally, the web application [1] communicates through the hypertext transfer protocol (HTTP) protocol, which involves HTTP web requests and HTTP responses. Applications servers use session objects [2] to store information about user sessions, including the session id, user id, profile id, user privileges, types, and any other data or localization needed for the application's business functions and smooth workflows. This session information is typically stored in a session table or session object on the application server side and is mapped to the session id sent from the client side. There are two main groups of vulnerabilities [3]–[6] in web applications: injection (technical vulnerabilities) and business logic vulnerabilities, which include broken access control (BAC) vulnerabilities. Technical vulnerabilities often result from insufficient input validation, allowing attackers to inject attacks such as cross-site scripting and structured query language (SQL) injection [7]. These attacks involve injecting malicious payloads into the application to perform malicious actions. Various technologies, such as web application firewalls (WAF) [8], can help protect against these types of attacks. In contrast, logical attacks are more difficult to address. Despite their criticality and in contrast to input validation vulnerabilities, the logical attacks have very limited work to address.

Open web application security project (OWASP) reports that 94% of applications have vulnerabilities related to broken access control as the most discovered and reported through penetration testing [9]. Broken access control vulnerability raised from fifth to first place on OWASP Top 10 web application vulnerabilities list for 2021 [9]. Access control vulnerabilities occur when a user can circumvent the allocated permissions and rights. This usually leads to unauthorized access, privilege escalation, information leakage, and data integrity issues. These vulnerabilities might be caused by insecure coding or unprotected authentication and authorization procedures.

Common access control flaws include the following i) Allowed forced browsing to pages requires authentication for unauthenticated users, or allows access to a highly privileged page from users with inadequate rights; in this category, the vulnerability occurred when some private pages (e.g., back up files, operations, and configuration files) does not have proper authentication and authorization policies which allow an intruder to exploit this flaw and accesses those restricted pages or resources through directly accessing their uniform resource locator (URL) bypassing the intended but not properly implemented normal authentication and authorization procedure; ii) Another type occurs when some specific functions that only intended to be accessed from a specific user role (e.g., bank approver role) while the application implementation allows the other user (e.g., maker role) to call the approving function directly to process an item despite the authorization level; iii) Another critical broken access control category occurs when a standard low-privileged user can gain increased privileges, such as being able to act as an administrator by tricking the application into thinking that the user is an administrator, this can take place through multiple scenarios including manipulating user session parameters or accessing admin portal functions and escalate the user or the session rights; iv) Another type of access control vulnerability in HTTP request methods occurs when an application fails to restrict access to certain HTTP methods (e.g., PUT request method or DELETE request method) which allows the intruder to utilize the web method to manipulate the application integrity and confidentiality; v) Other types of access control violations can especially exist in multi-step operations or workflows where the access control rules may be applied on specific steps but are not applied consistently on other steps moreover it may occur when contextual access control rules do not take into consideration the steps execution order and prerequisites steps for each specific step, a common example for this vulnerability is not preventing the modification of the shopping cart items after order bill calculation; and vi) Another major category of broken access control vulnerabilities is what is commonly named insecure direct object reference (IDOR) [10] which occurs when an application exposes a reference to an internal object directly without validating the request authentication and authorization parameters, as an example, consider a web application that allows the user to retrieve his personal information saved at a specific system using his national identification number (NIN) through the following web application sample URL (e.g., *example.com/show_my_details/NIN/28889900001*); however, a malicious actor can modify the NIN parameter in the URL to view other user's personal information through typing the target user’s NIN (e.g., *example.com/show_my_details/NIN/28889900002*), this example can apply to other resources.

Identifying broken access control vulnerability requires a thorough knowledge and understanding of both the application's intended behavior of the application and the actual implementation of the application. This requires understanding the relation between inputs and outputs along with the intended data flow and correlating it with the level of resource permissions and user rights. There are multiple factors to be considered for application security to prevent the existence of such vulnerability like robust session management, and proper account permissions configuration, nevertheless, multiple frameworks are designed to help with having proper authentication and authorization in the application but it needs a thorough configuration to reach the intended result. The researchers who focused on detecting BAC vulnerability and detecting BAC exploits topic explored and developed some methods such as static and dynamic code analysis, enhancing the software development life cycle, nevertheless developed black box approaches to detect state violation attacks and malicious SQL queries that are used in exploiting the vulnerabilities.

This paper considers different vulnerability detection and exploit prevention for Brocken access control vulnerability. Our contributions are detailed as follows:

a. The study discusses various broken access control vulnerability and exploitation detection techniques, concepts, and architecture, along with their pros and cons.
b. A comprehensive comparison of exited broken access control vulnerability and exploitation detection techniques and concepts that are used.
c. We spotted some exciting research gaps were found in the detection and prevention techniques.

d. open research problems, challenges, and future research directions are discussed to provide adequate protection against logical attacks.
e. The study categorizes different broken access control vulnerability and exploitation detection methods according to techniques categories.

The rest of this paper is divided into five sections. In section 2, we explore various categories of existing countermeasures. Section 3 analyzes the relevant literature and its contribution and limitations. Section 4 includes a discussion of gaps and potential research directions. The paper concludes in section 5 along with outlining future work direction.

2. COUNTERMEASURES FAMILIES

Research in mitigating broken access control vulnerability risks can be categorized into three major categories. The first category focuses on preventing the creation of the vulnerability itself in the coding phase in addition to detecting and fixing the vulnerability through code review activities. The second category aims to detect the vulnerability in the running application and make the needed effort to mitigate it. The third category targets detecting and preventing the exploitation of existing broken access control vulnerabilities in the running applications.

The existing literature has proposed research to secure web applications from logic flaws during the different phases of the software life cycle. The researchers suggest a wide diversity of coding practices recommendations during the development stage (e.g., in [11], [12]) in addition to secure web frameworks suggested as in the following references [13]–[17] to address the security aspects from different perspectives, some frameworks functions are providing the developers with a set of tools to implement the security policies (e.g., [18], [19]) in development and architecture phases. Nevertheless, some frameworks functions exist to leverage the auditing and authorization capabilities. Source code review plays a crucial role in detecting security flaws in software systems including BAC vulnerabilities. Studies (e.g., [20]–[22]) found that source code review while utilizing automated source code review tools can effectively detect multiple security issues categories including broken access control. These studies emphasize the importance of incorporating source code review into the software development life cycle to improve the security of the application and strengthen it against potential exploitations.

Security testing is one of the most effective and practical techniques for identifying vulnerabilities in software solutions including broken access control vulnerabilities. Studies (e.g., in [22]–[27]) found that security testing can effectively find BAC vulnerabilities as it analyzes the behavior of the running system and evaluates the implementation of access control policies and applied controls and security levels. Other studies [28] discussed the use of security testing in combination with code review alongside threat modeling to identify BAC vulnerabilities in web applications; nevertheless, studies emphasize the value of the usage of automated security testing tools for effective and automated detection of broken access control vulnerabilities as it can provide a thorough assessment and analysis of the target application and help identify issues that may be overlooked by manual testing.

Limited research exists to prevent the exploitation of broken access control vulnerabilities in the running applications during runtime focusing on exploit and attack prevention instead of detecting the vulnerabilities. The research in this area used different techniques and methodologies to distingue the application's benign and malicious requests. Research in these areas aimed to learn application valid invariants and states to reject invalid states (e.g., [29]–[31]). This category of research aims to overcome the limitations in Blackbox testing and Whitebox testing such as including its requirements for cost and resources for applying the needed mitigations, in addition, some research in this category targeted to addresses the code access requirement constraint.

Research directions and approaches related to the usage of frameworks and coding can be considered ineffective by design as they require to be enforced and used at the early development stage as they cannot be used with running applications. They require the source code existence in case a refactoring approach is needed, it also only deals with the vulnerability existence, not the active exploitation prevention. Additionally, the research focused on static code analysis [32] has a major limitation as it requires the source code as well which is not valid in most cases since a wide percentage of the applications is closed source and source code owners are not willing to provide for multiple reasons, also it requires time to fix and deploy putting the whole effort on developers, and it is a technology and programing language highly dependent which make it inefficient in most of the environments which usually have diversity of applications and technologies and it still facing high false positive rates. In addition, it is considered a detection tool for the vulnerability existence, not the active exploitation prevention

Dynamic analysis solutions such as DetLogic [23] and LogicScope [24], along with similar models, aim to detect vulnerabilities in running applications that are limited by design. These methods are heavily dependent on the assessor's ability to efficiently crawl the application during the learning phase. As a result, any areas, pages, or functions not explored by the assessor will not be assessed or protected. Furthermore, it requires additional costs and resources to implement necessary fixes for the discovered vulnerabilities. There is no guarantee that closed-source providers and product manufacturers will rectify the issues in a timely and acceptable manner.

Other approaches exist as well to support defending against logical attacks from different specific perspectives such as the requirements engineering language for adaptive information security (RELAIS) approach [33]. RELAIS is designed to address a specific issue related to parameter tampering attacks, where it is not feasible to establish all defensive mechanisms during the design phase. RELAIS utilizes the knowledge gained during system runtime and failures in specific about the environment along with applying machine learning techniques like Bayesian classification, logistic regression, and input approximation techniques to only extract requirements and determine behavior to identify needed adaptation to be implemented later.

3. ATTACK AND EXPLOIT DETECTION TECHNIQUES

One of the approaches to mitigate the broken access control vulnerability risk is to detect and prevent the exploitation phase of the vulnerability. One of the methodologies to detect exploitation and anomalies is by learning the application specification and then identifying the deviations between the intended application behavior and its runtime behavior. Researchers utilized different data sources and parameters for performing the learning process in their proposed solutions and also used different algorithms and concepts to have better verdicts and overcome limitations.

Researchers presented BLack-bOx approach for detecting state violation attaCKs (BLOCK) [29], The BLOCK technique is about detecting types of broken access control attacks in web applications using a black-box method. In the context of this technique, the web application is viewed as a stateless system where the intended behavior of the system is inferred by examining how clients interact with the application. Web requests and responses are assessed at runtime using the identified invariants from the web request/response sequences and associated session variable values during normal system operation. A potential state violation attack is one that deviates from these invariants in either the request or the response. BLOCK researchers implement their techniques as proxy [34] between client and server to intercept the messages between client and web application server and collect the session data from session files in training mode, then they generate malicious traffic to evaluate the solution's effectiveness. Results show high false positive rates due to the following limitations; BLOCK only focuses on the relation between requests, responses, and session variables which lake of visibility on the persistent information stored in the database, the stored information in the database may be used to maintain the session across more complex scenarios and workflows and multiple different web sessions, additionally, BLOCK cannot capture indirect relations, BLOCK requires manual intervention to guarantee adequate manual learning and filter out false positives. BLOCK by design will consider all unvisited paths or built invariants as attacks which as well contribute to raising false positive rates. The solution did not take into consideration tasks that must be executed by humans such as workflow procedures [35].

The researchers introduced securing database from logic flaws in web application (SENTINEL) [30], SENTINEL is a black-box approach to detect SQL queries that violate the application's intended behavior and produce logic flaws, their methodology is composed of systematically extracting a set of invariants from observed SQL queries, responses, and session variables, they model the interactions between the web application and the database based on the principals of the extended finite state machine; they consider SQL queries that not have corresponding invariant as a potential attack and drop it. They implemented their solution through two components sensor and analyzer, the sensor collects traffic such as SQL quires its responses along with session variables and communicates with the analyzer, while the analyzer performs offline training by extracting SQL signatures and infers the set of invariants associated with signatures, In runtime, the analyzer evaluates incoming SQL queries and directs the sensor to block any violating queries, SENTINAL overcomes some of the limitations in BLOCK since it takes in consideration the persistent state in the database, additionally its visibility on SQL queries provides more capability in blocking attacks targeted database integrity. SENTINAL limitations include that the solution does not take into consideration NoSQL [36] database backend web applications and can only be applied to the traditional flat relational data model, moreover, it can only address traditional SQL queries that have the same patterns in different languages [37], moreover, can only be applied to specific web development languages and platforms, another limitation from the performance point of view that it introduces performance overhead in SQL response time because of the communication overhead between sensor and analyzer and the analysis time during which analyzer extracts SQL signature and evaluates the query, SENTINAL provides a slight enhancement on false positive rate comparing to BLOCK but still requires some additional techniques to suppress false positives. SENTINAL as well by design considers all unvisited paths or recorded invariants as attacks that also contribute to raising false positive rates, nevertheless, any change in the application structure or data layer will make the learned invariants invalid and require a new round of learning to avoid false positives and incorrect application wide blockage state.

Invariant detector (IVD) [31] is an approach for automatic learning and enforcement of authorization rules in online social networks [38], the solution approach is to block requests attempting to exploit vulnerabilities in the authorization logic of online social networks. The solution technique starts with the learning phase which can take place at the staging phase or testing or pre-release stages of the features in scope, IVD intercepts requests made by an online social network (OSN) to its database, and it stores the likely invariants as attributed graph model [39], nevertheless, IVD design allows it to adapt automatically to OSN changes by continuously learning invariants to minimize manual learning requirements, the IVD composed of three components, request sampler which handles the database queries, the second component the invariant inference which uses the first component output for offline learning, the last component is the invariant checker which is responsible for validating the requests and block the malicious request. IVD work limitation is that it is specific for online social networks and requires manual intervention through writing manual customized rules to minimize potential false positives and false negatives which makes it a challenge to apply it generally in other applications and other domains, IVD's scope is to verify only on write access control policies and their relevant vulnerabilities and exploitations an example for that is that only "Alice's friends can post to Alice's profile"; IVD can contribute to confidentiality indirectly through blocking exploits that can result in "friend" maliciously created a relation between two users, which allows the attacker to collect information from some users as a friend role, IVD, however, does not validate access control on reads and, thus, cannot completely enforce data confidentiality [40], the authors as well planned to overcome some limitation in their future work by extending the approach to cover the more complex invariants. The solution does not take into consideration NoSQL database backend web applications.

InteGuard [41] is an approach to tackle the particular threat surface related to logical flaws resulting from third-party services API integration, the research discussed the new threats associated with the integration process with external web services such as payment services and single-sign-on (SSO), the research aims to reach a solution for how to securely integrate different three parties involved, including the merchant service (integrator), the provider of the third party services such as PayPal and the web clients using the application. The InteGuard which acts as a proxy deployed to analyze the ingress and egress traffic of the service integrators web' site composed of three main components, a trace collector used to generate and label learning traffic, an internet content adaptation protocol (ICAP) server to intercept the direct communication between the integrator and the provider, both of the components analyze and extract the needed parameters and relations and analyzing responses scripts and hypertext markup language (HTML) content then send it to the security policy generator to extract the invariants and construct a finite state machine (FSM) that reflects the security policies for the integration. The ICAP server inspects HTTP messages that have been sent to the integrator's website to perform the global policy by checking and extracting elements of interest from the messages and checking their compliance with security policies. Policies can be tuned when false positives are discovered during the protection mechanism operation. Limitations of this approach include that it does not protect provider-side flaws while several attacks can only be detected at that side such as variants of the unauthorized login by auth. code (or token) redirection attack from [42]. InteGuard only analyzes the application and network traffic for invariants creation and loses the advantages of inferring invariants at the database layer [31] which instead offers more advantages by providing both comprehensive invariants and scalability. The solution did not take into consideration tasks that must be executed by humans such as workflow and multistep procedures [35] and does not enforce authorization policies or constraints. InteGuard primarily focuses on browser-based web applications and not on mobile-based application merchant platforms [43].

Swaddler [44] is an anomaly detection method for detecting workflow relevant attacks, Swaddler uses anomaly detection in the internal state of the web application to detect vulnerabilities. The model works in one of two modes, training and detection mode, the training phase is required to record the characteristics of normal events including the code execution paths linked to session variables, two components are used in the implementation, the sensor is an extension of the hypertext preprocessor (PHP) interpreter that analyze the ongoing state of an application, while the analyzer is an anomaly-based system that assesses the regularity of the application's state and anomaly score thresholds to distinguish between regular and anomalous values are created when the sensor has finished its processing, it invokes the original handler of the statement, passing and resume the application path execution. After creating the profiles, which means that the models have collected the needed info about normal events as well and the appropriate thresholds have been established, the system can work in the detection mode. During this mode, the system calculates anomaly scores and reports any anomalous states. One of the main Swaddler limitations that it is requires source code access which makes it technology and source code dependent along with that it necessitates the modification of the PHP engine to enable monitoring of the web application's execution flow and the need to access execution paths to monitor, which could present major practical deployment difficulties. Moreover, it is crucial to consider the performance trade-off of Swaddler. Moreover, Swaddler cannot detect categories of data flow violation [45] such as insecure direct object reference (IDOR) and types of broken access control (BAC). Follow-up research as in [46] considers the challenge of tuning in operational settings and a large number of false positives due to the usage of unsupervised learning.

Double Guard [47] is an intuition detection system (IDS) that targets to detect privilege escalation attacks, hijack future session attacks, injection attacks, and direct database attacks. Double guard is deployed in such a way as to monitor both users' requests to a web server and its subsequent corresponding database requests to the back-end to detect malicious activity, doable guard utilizes container-based web server architecture where each session to be assigned to a separate web server and isolated from other sessions and mapped to a single user to enable separating different information flows by its every session and allow linking it to corresponding database query or request. Limitations of double guard include the inability to detect some of the broken access control categories including that it cannot detect insecure direct object reference attacks (IDOR) since it focuses on query structure anomalies, moreover, it is designed to work on detection and not prevention mode [48], and does not employ incremental learning which requires the model learning phase to be reset in case of application new legitimate features added, additionally, there is a big challenge in scalability and performance management given that the proposed architecture which requires container image per each single user session. Double guard identifies user session per their internet protocol (IP) address which is not reliable given the users' clusters who uses proxy (e.g., corporates or countries) and mobile users who dynamically change their IP address.

TamperProof [49] is an inline defense tool deployed as a proxy between the client and server, the tool can be used to protect legacy against parameter tampering attacks. The solution approach is to infer and enforce field, and value constraints on each input submitted to the server dynamically, it analyzes each form generated by the server to extract the constraints enforced by HTML and JavaScript and record the constraints. validation takes place by rejecting any request to the server that does not satisfy the constraints corresponding to the page form used to submit the input. TamperProof also injects a hidden field into each web form an identifier referred to as patchID which is used to validate the requests and sequencing. The main limitation of the TamperProof tool that is it cannot provide protection applications that alter the client code of a web form and Asynchronous JavaScript and eXtensible Markup Language (AJAX) requests in addition to those written for Web 2.0 or Web 3.0 or any dynamic altering client code which make it almost unusable for most of the currently running applications; additionally, web pages which employed HTML and JavaScript obscuration will be another challenge.

PHP-Sensor [50] is a model that approaches to discover workflow violation attacks and cross-site scripting (XSS) attacks that target PHP applications. To perform the workflow violation attack detection, the model observes the sequences of HTTP requests/responses and their associated session variables while in offline mode to extract a specific set of axioms. This set of axioms is later applied to assess the HTTP request/response during online mod where If an HTTP request/response fails to comply with its corresponding axiom, it is considered a workflow violation attack within a PHP web application, additionally, it can detect XSS worms through monitoring HTTP web requests and responses. The deployment will utilize a web proxy which is responsible for the workflow violation filter process, it identifies workflow violation attacks by constructing the expected flow model of the web application by analyzing the communication between the web browser and the server. The PHP-Sensor defensive model, which detects workflow violation attacks, comprises two primary phases: the offline phase and the recognition phase. The proposed framework, PHP-sensor, detects workflow violations using a recognition phase that involves a set of axioms. Each HTTP web request key is linked to two specific types of axioms, while input/output pairs are linked to different types of axioms. Axioms are converted into estimation functions that yield true or false values based on whether input pairs satisfy them. The recognition phase authenticates input and output pairs by checking if the associated axioms are fulfilled, and if not, the HTTP request or HTML page is blocked. Limitations of the PHP-Sensor model include that it is technology dependent since it works only on PHP, in addition, limitations of the model are that it does not detect other categories of broken access control such as direct object reference attacks (IDOR) and unauthorized access to functions and privilege escalations, moreover, PHP-Sensor is not applicable to web applications developed on platforms such as AJAX. Additionally, the PHP-Sensor's capability to control complex restrictions within the database is limited. The authors of the solution also planned to overcome some of these limitations by considering other internal states of PHP web applications. Furthermore, they will focus on optimizing several methods to reduce the performance overhead caused by PHP-Sensor.

4. DISCUSSION AND RESEARCH DIRECTION

Exploitation prevention and detection current approaches and techniques for BAC have different working models and implantations which result in different value added and shortcomings. In order to infer the current techniques challenges and research directions, a thorough evaluation and comparison took place as illustrated in Tables 1 and 2 based on the following characteristics, false positives (FP), presetancy aware (PA) in which we evaluate if the technique takes into account the persistent state information in the application logic, workflows aware (WA) in which we evaluate if the technique takes into account the workflow BAC relevant attacks, manual intervention required (MIR) for false positive filtering, manual crawling is required (MCR) for the application functions as any non-manually learned functions is considered a false positive, source code required (SCR), performance noticeable overhead (PO), NoSQL aware (NSA) in which it can handle the persistency while the backend is non-relational NoSQL databases, specific web development languages and platforms (SDLP), IDOR in which we evaluate if the technique can detect IDOR,: broken access control privilege escalation (BACPE), business or domain specific (BDS), and blocking mode (BM) in which we evaluate if the technique can work in blocking mode and not only in detection mode with no blocking capabilities.

Table 1. Solutions comparison part 1

	Solution	Data Source	FP	PA	WA	MIR	MCR	PO
1	BLOCK [29]	Requests, responses, and session variables	High	No	No	Yes	Yes	Average
2	SENTINEL [30]	SQL queries, responses, and session variables	Exist	Yes	Yes	Yes	Yes	Requires improvement
3	IVD [31]	Database requests and logs	Exist	Yes	Yes	Yes	Yes	Trade-off
4	InteGuard [41]	Ingress and egress traffic of the service integrators. Communication between the integrator and the provider. Trace Collector is used to generate and label learning traffic.	Exist	No	No	Yes	Yes	Average
5	Swaddler [44]	Code execution paths linked to session variables	High	No	Yes	Yes	Yes	Requires improvement
6	DoubleGuard [47]	User requests to a web server and its subsequent corresponding database requests to the back-end	Exist	No	No	Yes	Yes	Requires significant improvement
7	TamperProof [49]	It analyzes each form generated by the server to extract the constraints enforced by HTML and JavaScript and record the constraints	High	No	Yes (workflow)	No	No	Requires improvement
8	PHP-Sensor [50]	Observes the sequences of HTTP requests/responses and their associated session variables	Exist	No	Yes	Yes	Yes	Requires improvement

Table 2. Solutions comparison part 2

	Solution	NSA	SDLP	IDOR	BACPE	SCR	BDS	BM
1	BLOCK [29]	No	Yes	Yes	Yes	No	General	Blocking mode
2	SENTINEL [30]	No	Yes	Yes	Yes	No	General	Blocking mode
3	IVD [31]	No	Yes	Yes	Yes	No	Domain specific: online social networks (OSN) Enforce Integrity. Does not directly enforce confidentiality.	Blocking mode
4	InteGuard [41]	No	No	No	No	No	Third-party services APIs integration. Protects only the merchant side. No mobile application	Blocking mode
5	Swaddler [44]	No	Yes	No	No	Yes	Workflow relevant attacks only	Detection only
6	Double Guard [47]	No	No	No	Yes	No	Double Guard identifies user sessions per their IP	Detection only
7	TamperProof [49]	No	Yes	No	Yes	No	Not supporting AJAX, Web 2.0 or Web 3.0, or obfuscated HTML and JS	Blocking mode
8	PHP-Sensor [50]	No	Yes	No	No	Yes	Not supporting web applications developed on platforms such as AJAX	Blocking mode

Based on the thorough comparison as elaborated, we can group the current gaps and limitations along with the corresponding research direction as listed following:

a. The wide spectrum of the existing literature that aims to detect broken access control vulnerability and its sub-category such as insecure direct object reference have a serious limitation as those solutions by design requires application crawling through human and spiders and then considering all not crawled paths or recorded invariants as attacks, that strategy impacts model efficiency including striking the false positive rates, nevertheless, one of its results that any change in the application structure or data layer make the learning phase or learned invariants invalid and requires a new round of learning to avoid false Positives and incorrect application wide blockage state, research is required in this point to overcome this challenge through exploring the available machine learning and artificial intelligence techniques and algorithms that can provide a solution to this focal issue.

b. Most of the provided solutions are either very dependent and custom only to be operating on or protect specific technology in a matter of programming language used or technology used such as PHP applications only or specific databases types which not able to protect other types of databases NoSQL database backend web applications. Research is required to provide less technology dependent and more generalized solutions that can cover the wide spectrum of technologies and implementation diversifications.
c. Some solutions require the source code access during learning and blocking mode, providing the source code which is not applicable in most cases since a wide percentage of the applications is closed source and source code owners are not willing to provide for multiple reasons, nevertheless, it makes the solution very dependent on the language used, research is required to overcome this challenge to provide less source code dependent techniques.
d. Some of the solutions provided are very dependent on specific business type or services and needs further research to be generalized to be a more generic solution for securing against broken access control attack.
e. Most of the provided solutions are not capable of detecting the severe insecure direct object reference category, research is required to close this gap and find innovative and effective techniques to safeguard against the vulnerability.
f. Horizontally, deeper research is required to overcome the performance impact on most of the solutions to be practically visible in enterprises and to enhance false positive and false negative rates.

5. CONCLUSION AND FUTURE WORK

The existent solutions cannot be considered reliable and comprehensive enough to provide an adequate level of protection against critical broken access control attacks including some of its subcategories such as insecure direct object reference. This research provides a thorough analysis of the current literature, identifying current gaps and limitations. This paper suggests research areas and directions for tackling this critical issue.

Future work can be conducted to evaluate machine learning and artificial intelligence techniques to develop a methodology and model that can be an effective and efficient solution to detect and prevent broken access control attacks. The target model implementation should ensure that any identified limitations issues and gaps have been addressed. The research will also assess the principles of incremental learning, a process where the model continues to learn and adapt from new data, thereby enhancing its predictions or decisions over time.

REFERENCES

[1] R. T. Fielding and R. N. Taylor, “Principled design of the modern web architecture,” *ACM Transactions on Internet Technology*, vol. 2, no. 2, pp. 115–150, May 2002, doi: 10.1145/514183.514185.

[2] K. Gutzmann, “Access control and session management in the HTTP environment,” *IEEE Internet Computing*, vol. 5, no. 1, pp. 26–35, 2001, doi: 10.1109/4236.895139.

[3] A. Doupé, M. Cova, and G. Vigna, “Why Johnny can’t pentest: an analysis of black-box web vulnerability scanners,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6201, Springer Berlin Heidelberg, 2010, pp. 111–131.

[4] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna, “Toward automated detection of logic vulnerabilities in web applications,” in *Proceedings of the 19th USENIX Security Symposium*, 2010, pp. 143–160.

[5] J. Kaur and U. Garg, “State-of-the-art survey on web vulnerabilities, threat vectors, and countermeasures,” in *Studies in Computational Intelligence*, vol. 1007, Springer Singapore, 2022, pp. 3–17.

[6] G. Deepa and P. S. Thilagam, “Securing web applications from injection and logic vulnerabilities: Approaches and challenges,” *Information and Software Technology*, vol. 74, pp. 160–180, Jun. 2016, doi: 10.1016/j.infsof.2016.02.005.

[7] W. G. J. Halfond and A. Orso, “Amnesia: analysis and monitoring for neutralizing SQL-injection attacks,” in *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, Nov. 2005, pp. 174–183, doi: 10.1145/1101908.1101935.

[8] S. V Pingale and S. R. Sutar, “Analysis of web application firewalls, challenges, and research opportunities,” in *Lecture Notes in Electrical Engineering*, vol. 783, Springer Singapore, 2022, pp. 239–248.

[9] OWASP Top 10:2021, “A01:2021 – broken access control,” *OWASP*. https://owasp.org/Top10/A01_2021-Broken_Access_Control/#references (accessed Jun. 19, 2022).

[10] A. K. Shrestha, P. S. Maharjan, and S. Paudel, “Identification and illustration of insecure direct object references and their countermeasures,” *International Journal of Computer Applications,* vol. 114, no. 18, pp. 39–44, Mar. 2015, doi: 10.5120/20082-2148.

[11] Y. Ndiaye, O. Barais, A. Blouin, A. Bouabdallah, and N. Aillery, “Requirements for preventing logic flaws in the authentication procedure of web applications,” in *Proceedings of the ACM Symposium on Applied Computing*, Apr. 2019, pp. 1620–1628, doi: 10.1145/3297280.3297438.

[12] OWASP, "Authorization cheat sheet," *OWASP*. https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html (accessed Jan. 29, 2023).

[13] N. Swamy, B. J. Corcoran, and M. Hicks, "Fable: a language for enforcing user-defined security policies," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*, May 2008, pp. 369–383, doi: 10.1109/SP.2008.29.

[14] S. Chong, K. Vikram, and A. C. Myers, "SIF: enforcing confidentiality and integrity in web applications," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, 2007, pp. 1–16.

[15] L. Jia *et al.*, "AURA: a programming language for authorization and audit," in *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, Sep. 2008, pp. 27–38, doi: 10.1145/1411204.1411212.

[16] K. Vikram, A. Prateek, and B. Livshits, "Ripley: automatically securing web 2.0 applications through replicated execution," in *Proceedings of the 16th ACM conference on Computer and communications security*, Nov. 2009, pp. 173–186, doi: 10.1145/1653662.1653685.

[17] J. Morgenstern and D. R. Licata, "Security-typed programming within dependently typed programming," in *Proceedings of the 15th ACM SIGPLAN International Conference on Functional programming*, 2010, pp. 169–180, doi: 10.1145/1863543.1863569.

[18] A. Yip, X. Wang, N. Zeldovich, and M. F. Kaashoek, "Improving application security with data flow assertions," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, Oct. 2009, pp. 291–304, doi: 10.1145/1629575.1629604.

[19] J. Yang, T. Hance, T. H. Austin, A. Solar-Lezama, C. Flanagan, and S. Chong, "Precise, dynamic information flow for database-backed applications," in *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Jun. 2016, vol. 51, no. 6, pp. 631–647, doi: 10.1145/2908080.2908098.

[20] M. Ghorbanzadeh and H. R. Shahriari, "ANOVUL: detection of logic vulnerabilities in annotated programs via data and control flow analysis," *IET Information Security*, vol. 14, no. 3, pp. 352–364, May 2020, doi: 10.1049/iet-ifs.2018.5615.

[21] M. Monshizadeh, P. Naldurg, and V. N. Venkatakrishnan, "MACE: detecting privilege escalation vulnerabilities in web applications," in *Proceedings of the ACM Conference on Computer and Communications Security*, Nov. 2014, pp. 690–701, doi: 10.1145/2660267.2660337.

[22] B. Zhang, J. Li, J. Ren, and G. Huang, "Efficiency and effectiveness of web application vulnerability detection approaches: a review," *ACM Computing Surveys*, vol. 54, no. 9, pp. 1–35, Dec. 2022, doi: 10.1145/3474553.

[23] G. Deepa, P. S. Thilagam, A. Praseed, and A. R. Pais, "DetLogic: a black-box approach for detecting logic vulnerabilities in web applications," *Journal of Network and Computer Applications*, vol. 109, pp. 89–109, May 2018, doi: 10.1016/j.jnca.2018.01.008.

[24] X. Li and Y. Xue, "LogicScope: automatic discovery of logic vulnerabilities within web applications," in *ASIA CCS 2013 - Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, May 2013, pp. 481–486, doi: 10.1145/2484313.2484375.

[25] X. Li, X. Si, and Y. Xue, "Automated black-box detection of access control vulnerabilities in web applications," in *Proceedings of the 4th ACM Conference on Data and Application Security and Privacy*, 2014, pp. 49–60, doi: 10.1145/2557547.2557552.

[26] G. Pellegrino and D. Balzarotti, "Toward black-box detection of logic flaws in web applications," *Proceedings 2014 Network and Distributed System Security Symposium*, 2014, doi: 10.14722/ndss.2014.23021.

[27] G. Deepa, P. S. Thilagam, F. A. Khan, A. Praseed, A. R. Pais, and N. Palsetia, "Black-box detection of XQuery injection and parameter tampering vulnerabilities in web applications," *International Journal of Information Security*, vol. 17, no. 1, pp. 105–120, Jan. 2018, doi: 10.1007/s10207-016-0359-4.

[28] S. C, J. G, R. V. P, and E. P R L, "PROP - patronage of PHP web applications," *International Journal of Computer Science and Information Technology*, vol. 7, no. 2, pp. 111–125, Apr. 2015, doi: 10.5121/ijcsit.2015.7210.

[29] X. Li and Y. Xue, "BLOCK: a black-bOx approach for detection of state violation attacks towards web applications," in *ACM International Conference Proceeding Series*, Dec. 2011, pp. 247–256, doi: 10.1145/2076732.2076767.

[30] X. Li, W. Yan, and Y. Xue, "SENTINEL: securing database from logic flaws in web applications," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, Feb. 2012, pp. 25–36, doi: 10.1145/2133601.2133605.

[31] P. Marinescu, C. Parry, M. Pomarole, Y. Tian, P. Tague, and I. Papagiannis, "IVD: automatic learning and enforcement of authorization rules in online social networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 1094–1109, doi: 10.1109/SP.2017.33.

[32] R. Croft, D. Newlands, Z. Chen, and M. A. Babar, "An empirical study of rule-based and learning-based approaches for static application security testing," in *Proceedings of the 15th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Oct. 2021, pp. 1–12, doi: 10.1145/3475716.3475781.

[33] T. T. Tun *et al.*, "Requirements and specifications for adaptive security," in *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems*, May 2018, pp. 161–171, doi: 10.1145/3194133.3194155.

[34] N. Weaver, C. Kreibich, M. Dam, and V. Paxson, "Here be web proxies," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8362, Springer International Publishing, 2014, pp. 183–192.

[35] L. Compagna, D. R. dos Santos, S. E. Ponta, and S. Ranise, "Aegis: automatic enforcement of security policies in workflow-driven web applications," in *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, Mar. 2017, pp. 321–328, doi: 10.1145/3029806.3029813.

[36] M. Stonebraker, "SQL databases v. NoSQL databases," *Communications of the ACM*, vol. 53, no. 4, pp. 10–11, Apr. 2010, doi: 10.1145/1721654.1721659.

[37] S. Wen *et al.*, "Toward exploiting access control vulnerabilities within MongoDB backend web applications," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Jun. 2016, vol. 1, pp. 143–153, doi: 10.1109/COMPSAC.2016.207.

[38] A. Aldo Tenis and R. Santhosh, "Challenges and security issues of online social networks (OSN)," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 68, Springer Singapore, 2022, pp. 703–709.

[39] S. Sakr, S. Elnikety, and Y. He, "G-SPARQL: a hybrid engine for querying large attributed graphs," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, Oct. 2012, pp. 335–344, doi: 10.1145/2396761.2396806.

[40] E. Elnikety, "Comprehensive and practical policy compliance in data retrieval systems," Ph.D. dissertation, Department of Computer Science, Saarland University, Saarbrücken, 2019.

[41] L. Xing, Y. Chen, X. Wang, and S. Chen, "InteGuard: toward automatic protection of third-party web service integrations," *Proceeding of the 20th Network and Distributed System Security Symposium*, 2013.

[42] C. Bansal, K. Bhargavan, A. Delignat-Lavaud, and S. Maffeis, "Discovering concrete attacks on website authorization by formal analysis," *Journal of Computer Security*, vol. 22, no. 4, pp. 601–657, Apr. 2014, doi: 10.3233/JCS-140503.

[43] W. Yang *et al.*, "Show me the money! Finding flawed implementations of third-party in-app payment in android apps," *Network and Distributed System Security Symposium*, 2017, doi: 10.14722/ndss.2017.23091.

[44] M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna, “Swaddler: an approach for the anomaly-based detection of state violations in web applications,” in *Recent Advances in Intrusion Detection*, vol. 4637, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 63–86.
[45] M. Ghorbanzadeh and H. R. Shahriari, “Detecting application logic vulnerabilities via finding incompatibility between application design and implementation,” *IET Software*, vol. 14, no. 4, pp. 377–388, Aug. 2020, doi: 10.1049/iet-sen.2019.0186.
[46] I. Jana and A. Oprea, “AppMine: behavioral analytics for web application vulnerability detection,” in *Proceedings of the ACM Conference on Computer and Communications Security*, Nov. 2019, pp. 69–80, doi: 10.1145/3338466.3358923.
[47] M. Le, A. Stavrou, and B. B. Kang, “DoubleGuard: detecting intrusions in multitier web applications,” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 512–525, Jul. 2012, doi: 10.1109/TDSC.2011.59.
[48] N. Agarwal and S. Z. Hussain, “A closer look at intrusion detection system for web applications,” *Security and Communication Networks*, vol. 2018, pp. 1–27, Aug. 2018, doi: 10.1155/2018/9601357.
[49] N. Skrupsky, P. Bisht, T. Hinrichs, V. N. Venkatakrishnan, and L. Zuck, “TamperProof: a server-agnostic defense for parameter tampering attacks on web applications,” in *Proceedings of the third ACM conference on Data and application security and privacy*, Feb. 2013, pp. 129–140, doi: 10.1145/2435349.2435365.
[50] S. Gupta and B. B. Gupta, “PHP-sensor: a prototype method to discover workflow violation and XSS vulnerabilities in PHP web applications,” in *Proceedings of the 12th ACM International Conference on Computing Frontiers*, May 2015, pp. 1–8, doi: 10.1145/2742854.2745719.

BIOGRAPHIES OF AUTHORS

Ahmed Anas is the head of the Cyber Security Governance Unit at The Central Bank of Egypt, before joining CBE, Ahmed was acting as EFG Holding and Valu CISO. Ahmed is an industry and cybersecurity expert and public speaker. Ahmed has extensive experience in creating and implementing cybersecurity strategies, designing, implementing, and assessing infrastructure and applications cybersecurity. His areas of experience also include governance, risk management and assessment, compliance, incident handling and response, penetration testing, Red Teaming, and SOC implementation. Ahmed earned his master’s degree in computer science from Cairo University with a thesis in the field of cyber security, Ahmed is an ISC² certified information systems security professional, EC-Council Certified Instructor, and Ethical Hacker, and obtained GIAC web application penetration tester and ISACA COBIT 2019 Foundation certificates. Ahmed is Ph.D. student in cyber security and AI. He can be contacted at email: ahmed.anas@pg.fci-cu.edu.eg.

Salwa Elgamal is a professor at the Computer Science Department, Faculty of Computers and Information, Cairo University. Vice dean of environmental development and community service, Faculty of Computers and Information, Cairo University. Head of Computer Science Department, Faculty of Computers and Information, Cairo University. Research interests include security fields and parallel computing. She can be contacted at email: s.elgamal@fci-cu.edu.eg.

Basheer Youssef is a teacher in the Computer Science Department, Faculty of Computers and Artificial Intelligence, Cairo University. He was awarded his Ph.D. in cryptography from the Faculty of Computers and Information, at Cairo University, in 2011. He was awarded his M.Sc. in cryptography from the faculty of computers and information, Cairo University, 2005. The B. Sc is awarded from the computer science department, faculty of computers and information, Cairo University, 2000. Research interests include algorithms and security fields. He can be contacted at email: basheer@fci-cu.edu.eg.