Algorithm 6 The Pseudocode for Extracting Processes

INPUT: the web application first node C_0
the web application Graph edges E
OUTPUT: the web application graph process P as a set of web application process

```
Begin
Let P_0 = ∅; // set of web application processes
Let StartEdge= ∅; // set of graph edge that begin from C_0
StartEdge= ExtractGraphEdges(C_0) //Extract all edges from C_0
EndPoint = ExtractEndPoint(StartEdge) //Extract the end point of edges
if (ExtractProcess(EndPoint, E) ≠ null) then
    return P=E+ExeractProcess(Endpoint, E); for any edge E ∈ StartEdge
else
    return E;
end if
end
```

the first node and the end node of the process are the same and the process length is greater than two. (If there is a return to the passed node in the process and the created loop length is more than two, this is a business process).

All processes in the graph from the initial node (the application initial page) to the identified final nodes, as well as the processes of their initial node and the final node are the same; and all of them are the application business processes. Algorithm 7 shows the pseudocode for identifying the application business process. In line 4, the application business processes are extracted. In line 5, the final nodes of the graph are extracted. In line 7, the processes that start with the initial node and end with the final node are identified as the business process are stored in the variable BP. In line 9, the processes with repeated nodes are detected and in line 11, among the detected processes, if their initial node and their final node are the same and their length is greater than two, they are added to the variable BP as the business process.

4 EXPERIMENTAL RESULTS

The testbed used in this section is a network consisting of a web server (test target) and two clients (BLProM system and legal user). The web server and the clients are loaded on a virtual machine. The web server and the clients' profiles are shown in Table 2.

The web applications listed in Table 3 are installed on the web server (test target) and then we plan to identify the business layer of the web applications.

The legal user first starts using the selected web applications. The user crawls all permitted parts of the web application. HTTP traffic of the legal user is given to BLProM as its input.

5 EVALUATION

BLProM's goal is to identify the business layer of the web application. We can identify business logic vulnerability by identifying the business layer of the web application. BLProM detects the business processes of the web application. Identifying business processes is the main step in dynamic security testing of the web application in the business layer.

We compare BLProM with OWASP ZAP. The OWASP Zed Attack Proxy (ZAP) is a free web scanner. It scans web applications and automatically finds some security vulnerabilities. ZAP is the only free web scanner that has API for extracting web application graph. The web application pages are graph nodes and the relations among pages are shown as graph edges. The main difference between BLPRoM and ZAP is in detecting similar pages. ZAP cannot detect similar pages in the web application but BLPRoM can. ZAP's graph only shows the relation among scanned pages but BLProM generates the optimal graph. About the accuracy of the generated graph, both BLProM and ZAP are the same.

To evaluate the proposed approach, we first show the accuracy of clustering by the following criteria:

- True Positive: Samples that fit well into their correct clusters.
- False Positive: Samples that fit in a cluster that do not belong to that cluster.
- False Negative: Samples that do not fit in a cluster but they belong to that cluster.
- Recall: It is calculated by the following formula:

$$recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- Precision: It is calculated by the following formula:

Table 2. Testbed Profiles.

Web server (test target)	CPU: Pentium dual core-2.20 GHZ OS: windows 8.1 VMware cpu: 1GHZ VMware RAM: 1G VMware OS: windows 7
Client (BLProM machine)	CPU: Intel corei7 2.20 GHZ OS: windows 8.1 VMware cpu: 1GHZ VMware RAM: 1G VMware OS: windows 7
Client (legal user)	CPU: Pentium dual core i3-3210 GHZ OS: windows 7 VMware cpu: 1GHZ VMware RAM: 1G VMware OS: windows 7

Table 3. Selected Web Applications for Evaluation.

Web application	**Description**
TomatoCart-1.1.8.6.1	e-commerce
osCommerce-2.3.4	e-commerce
WackoPicko	Web application for Sharing picture

Algorithm 7 The Pseudocode for Identifying the Application Business Processes

INPUT: the web application navigation graph $< C_0, C, E >$
the web application Graph edges E
OUTPUT: the web application graph business process PB as a set of web application process

Begin
Let $P = \varnothing$; // set of web application processes
Let $F = \varnothing$; //set of web application final nodes
P= ExtractProcess; //Extract processes in the web application
F= ExtractFinalNodes; //Extract Final nodes in the web application
for $i : 1 \ldots k$ **do**
    if P_k start by C_0 and ends by F **then**
        $BP = BP + P_k$
    end if
end for
R=ExtractProcessWithRepeatedNodes(P); //Extract Process with Repeated Nodes
for $j : 1 \ldots m$ **do**
    if R_j has same initial and end node and length(Rj) $\geqq$2 **then**
        $BP = BP + R_j$
    end if
end for
return BP;
end

Table 4. The Clusters of Selected Web Application Pages Evaluation.

Criteria \ Web application	WackoPicko	Tomatocart	osCommerce
#samples	89	150	210
#clusters	29	66	40
true positive	65	146	205
false positive	24	4	5
false negative	23	3	4
recall	0.74	0.98	0.98
precision	0.73	0.97	0.98
f-measure	0.73	0.98	0.98

Table 5. Comparing the Proposed Approach With OWASP ZAP in Scanning WackoPicko.

WackoPicko			
Criteria \ Approaches	Proposed approach	OWASP ZAP	Percentage of improvement compared to OWASP ZAP
# HTTP Message	89	89	–
# Graph Nodes	22	89	75.2
#Graph Edges	48	270	82.2
# process (P)	12	48	75
Average edge in each process ($\bar{E}$)	4	46	91.3
Average edges in all processes (P*$\bar{E}$)	48	2208	97.8
#business processes	10	NA	–

$precision = \frac{TruePositive}{TruePositive+FalsePositive}$

- F-Measure: It is calculated by the following formula:
  $F-Measure = \frac{2*recall*precision}{TruePositive+FalsePositive}$

The value of these criteria for the ClusteringWebPages algorithm (Algorithm 3) is shown in Table 4. In the first row, #samples shows the total number of web pages in HTTP traffic extracted from the ExtractWebPages algorithm in Algorithm 1. In the second row, #clusters shows the total number of clusters extracted from the ClustreringWebPages algorithm in Algorithm 3. In the next rows, the criteria listed above are calculated for each web application.

To evaluate the proposed approach, we compare the output of BLProM with the scanning output of OWASP ZAP for selected web applications. BLProM output for WackoPicko is shown in Table 5. #graph nodes shows the total number of clusters extracted from the ClustreringWebPages algorithm in Algorithm 3. #graph edges is the total number of edges extracted from the ExtractGraphEdges algorithm in Algorithm 4. #process is the total number of paths from the first node extracted from the ExtractProcess algorithm in Algorithm 6. Average edge in each process ($\bar{E}$) is calculated by the sum of edges of each process divided by the total number of processes. Average edge in all process is calculated by the product of the total number of processes (P) in average edges of each process ($\bar{E}$). #business processes is the total number of business processes in the web application.

Existing values in the first column of Table 5 are the output obtained from the proposed approach. The values in the second column are the scanning output of ZAP. The third column shows the percentage of scanning improvement of our proposed approach compared to the ZAP scan. The results of the table indicate that the ZAP scan is a non-smart scan. As a result of this non-intelligent scan, ZAP is not able to identify business layer vulnerabilities.

BLPRoM output for osCommerce is shown in Table 6. The third column shows the percentage of scanning improvement compared to ZAP scanning. It is observed that web application scanning is improved by identifying the web application business layer.

Table 6. Comparing the Proposed Approach With OWASP ZAP in Scanning osCommerce.

osCommerce			
Approaches / Criteria	Proposed approach	OWASP ZAP	Percentage of improvement compared to OWASP ZAP
# HTTP Message	170	170	–
# Graph Nodes	40	170	76.4
#Graph Edges	66	379	82.5
# process (P)	23	17	26
Average edge in each process ($\bar{E}$)	3	113	97.3
Average edges in all processes (P*$\bar{E}$)	69	1921	96.4
#business processes	18	NA	–

Table 7. Comparing the Proposed Approach With OWASP ZAP in Scanning TomatoCart.

TomatoCart			
Approaches / Criteria	Proposed approach	OWASP ZAP	Percentage of improvement compared to OWASP ZAP
# HTTP Message	150	150	–
# Graph Nodes	66	150	56
#Graph Edges	87	410	78.7
# process (P)	31	39	20.5
Average edge in each process ($\bar{E}$)	4	101	96
Average edges in all processes (P*$\bar{E}$)	156	3131	95
#business processes	30	NA	–

BLPRoM output for TomatoCart is shown in Table 7. The third column shows the percentage of scanning improvement of our proposed approach compared to ZAP scanning.

Table 8 shows the average of the proposed approach, the average of OWASP ZAP and the average percentage of improvement in the scanning of selected web applications. For example, in the "Average edges in all processes" benchmark, our approach has been improved by about 96 percent compared to OWASP ZAP.

According to the results presented in this table, can be observed that BLProM has improved web application scanning. BLProM is aware of web application business processes. By identifying the web application business layer, web scanners can detect business layer vulnerabilities.

6 CONCLUSION

Business logic vulnerabilities are strong vulnerabilities that compromise web application security. Web scanners cannot detect business logic vulnerabilities because they are unable to understand business logic of the web application. For detecting business logic vulnerabilities, the business logic of the web application needs to be understood. Therefore, these vulnerabilities are specific to the application and difficult to identify.

In this paper, we proposed BLProM, a black-box approach for detecting business processes of the web application. BLProM aims to ease detecting business logic vulnerabilities. BLPRoM output is used as input in dynamic security testing of the web applications in the business layer in order to detect business logic vulnerabilities. BLProM consists of two main steps:

1- extracting user navigation graph
2- Detecting web application business processes

At the lab, we scanned three web applications by BLProM and OWASP ZAP, an open-source web application. We showed that BLProM improved scanning about %96 compared to OWASP ZAP. BLProM improved the scanning of web applications because it clusters web pages and prevents scanning similar web

Table 8. Comparison of the Average of the Proposed Approach and the Average of ZAP Approach in the Scanning of Selected Web Applications.

Average of selected web application			
Criteria \ Approaches	Proposed approach	OWASP ZAP	Percentage of improvement compared to OWASP ZAP
# Graph Nodes	42.6	136.3	69.2
#Graph Edges	67	353	81.1
# process (P)	20	36.6	40.5
Average edge in each process ($\bar{E}$)	3.6	86.6	94.8
Average edges in all processes (P*$\bar{E}$)	91	2420	96.4

application pages.

References

[1] Common vulnerabilities and exposures. https://cve.mitre.org/cve/cve.html.

[2] 2015 ITRC. Identity Theft Resource Center Breach Report Hits Near Record High in 2015. http://www.idtheftcenter.org/ITRC-Surveys-Studies/2015databreaches.html, Accessed Feb, 2017.

[3] G. Pellegrino and D. Balzarotti. Toward Black-Box Detection of Logic Flaws in Web Applications. In *Network and Distributed System Security symposium 2014 (NDSS2014)*, 2014. doi:10.14722/ndss.2014.23021.

[4] Testing for business logic, OWASP. https://www.owasp.org/index.php/Testing_for_business_logic, Accessed Feb, 2017.

[5] D. Balzarotti, M. Cova, V. Felmetsger, and G. Vigna. Multi-module vulnerability analysis of web-based applications. In *Proceedings of the 14th ACM conference on Computer and communications security*, page 25–35. ACM, 2007. doi:10.1145/1315245.1315250.

[6] A. Doupé, B. Boe, C. Kruegel, and G. Vigna. Fear the EAR: discovering and mitigating execution after redirect vulnerabilities. In *Proceedings of the 18th ACM conference on Computer and communications security*, page 251–262. ACM, 2011. doi:10.1145/2046707.2046736.

[7] V. Felmetsger, L. Cavedon, C. Kruegel, and G. Vigna. Business Logic Attacks – Bots and BATs, OWASP, 2009. In *USENIX Security Symposium*, 2010.

[8] E. Chai. Business Logic Attacks – Bots and BATs, OWASP, 2009. https://www.owasp.org/images/a/aa/OWASP_Cincinnati_Jan_2011.pdf, Accessed Feb. 2017.

[9] X. Li and Y. Xue. BLOCK: a black-box approach for detection of state violation attacks towards web applications. In *Proceedings of the 27th Annual Computer Security Applications Conference*, page 247–256, 2011. doi:10.1145/2076732.2076767.

[10] M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna. Swaddler: An Approach for the Anomaly-Based Detection of State Violations in Web Applications. In *Giovanni*, pages 63–86. Springer, Berlin, Heidelberg, 2007. ISBN 978-3-540-74319-4. doi:10.1007/978-3-540-74320-0_4.

[11] 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*, page 25–36, 2012. doi:10.1145/2133601.2133605.

[12] M. Alidoosti and A. Nowroozi. BLDAST: business-layer Dynamic Application Security Tester of the web application in order to detect web application vulnerabilities against flooding DoS attacks. In *Iran Society of Cryptology Conference, shiraz, Iran.* in Persian, 2017.

[13] M. Alidoosti, A. Nowroozi, and A. Nickabadi. Evaluating the Web-Application Resiliency to Business-Layer DoS Attacks. *ETRI Journal*, 2019. doi:10.4218/etrij.2019-0164.

[14] M. Alidoosti and A. Nowroozi. BLTOCTTOU: business-layer dynamic application security tester of the web application in order to detect web application vulnerabilities against Race Condition attacks. In *Computer Society of Iran Conference, Tehran, Iran.* in Persian, 2018.

[15] M. Alidoosti and A. Nowroozi. BL-ProM: Business-layer process miner of the web application. In *ISCISC*, pages 1–6. IEEE, 2018. ISBN 978-1-5386-7582-3. doi:10.1109/ISCISC.2018.8546899.

[16] V. Crescenzi, P. Merialdo, and P. Missier. Clustering Web pages based on their structure. *Data & Knowledge Engineering*, 54(3):279–299, 2005. doi:10.1016/j.datak.2004.11.004.

Mitra Alidoosti received her B.S. and M.Sc. degrees in computer engineering from the Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran, in 2009 and 2012, respectively. Currently, she is working toward the Ph.D. degree in computer engineering at Malek-e-Ashtar University of Technology, Tehran, Iran. Her research interests are computer network security, VoIP and SIP security, and web-application security.

Alireza Nowroozi is a freelance consultant who advises government and private-sector-related industries on information technology. He has four-year experience as an academic staff member and an IT post-doctoral position with Sharif University of Technology, Tehran, Iran. He is a specialist in artificial intelligence, cognitive science, software engineering, and IT security. Besides, he is a co-founder of four IT startups.

Ahmad Nickabadi received his B.S. degree in computer engineering in 2004, and the M.Sc. and Ph.D. degrees in artificial intelligence in 2006 and 2011, respectively, from Amirkabir University of Technology, Tehran, Iran. He is currently an Assistant Professor in the Department of Computer Engineering, Amirkabir University of Technology. His research interests include statistical machine learning and soft computing.