

Holistic Web Application Security Visualization for Multi-Project and Multi-Phase Dynamic Application Security Test Results

FERDA ÖZDEMİR SÖNMEZ¹, (Member, IEEE), AND BANU GÜNEL KILIÇ², (Member, IEEE)

¹Institute for Security Science and Technology, Imperial College, South Kensington Campus London, London SW7 2AZ, U.K.

²Informatics Institute Middle East Technical University, Üniversiteler Mahallesi, 06800 Ankara, Turkey

Corresponding author: Ferda Özdemir Sönmez (f.ozdemir-sonmez@imperial.ac.uk)

ABSTRACT As the number of web applications and the corresponding number and sophistication of the threats increases, creating new tools that are efficient and accessible becomes essential. Although there is much research concentrating on network security visualizations, there are only a few studies considering the web application vulnerabilities' possible visualization options. Consequently, to fill this gap, this research centers around a novel perception configuration to improve web application vulnerability monitoring. This study forms a generic data structure based on data sources that might be readily associated and commonly available for the majority of the web applications. The primary contribution of this study is a new dashboard tool for visualizing dynamic application security test results. Another contribution is the metrics/measures that the tool presents. The paper also describes a validation study in which participants answered quiz questions upon using the tool prototype. For the case study, sample data has been generated using the OWASP ZAP scanner tool and a prototype has been implemented to be used for validation purposes. This study allows the investigation of fifty metrics/measures for the multi-project/phase environment that enhances its benefits if the user aims to monitor a series of analyses' results and the changes between them for more than one web project.

INDEX TERMS Computer security, information security, visualization, software engineering, web and internet services, dynamic application security testing, DAST, black-box test.

I. INTRODUCTION

The number of web-based applications is increasing each year. Although, there are no statistics on the number of existing web applications in the world, the number of domain names was around 367 million as of the first quarter of 2020. From one point of view, each of these domains might be considered as a web application, either static or dynamic.

Risk control and risk assessment are constant challenges for the software project management domain. Demir [1] surveyed on project management challenges with 78 participants. The results showed that approximately one in every four projects had problems in the security and risk control area. Web-based application architecture has been mainstream in medium and large size enterprises since many of the enterprise web applications are integrated within each other and with other enterprise systems. However, web applications are prone to continuous updates due to continuous changes

in the requirements and added functionalities. The adoption of web applications is increasing, yet these applications are commonly developed in an ad-hoc manner, without properly understanding the reliability and security requirements [2].

Security vulnerabilities are weaknesses which can be exploited by a threat factor, such as an attacker. The reason for focusing on web applications in this study is the increasing number of vulnerabilities and attacks, and the small number of visualization studies on this subject. Fig. 1 shows the trend of the Open Web Application Security Project (OWASP) top ten vulnerabilities between 2016 and 2019 [3].

Fortunately, security analyses and protection techniques are also improving. Doing only manual analyses and traditional tests are not sufficient. Thus, over the years, many automated analysis tools have been developed for efficient security checks. Some of these tools make white-box analyses, which are called static code analyzers. Static code analyzers use application code, resource, and configuration files as the analysis source. They are more suitable to be used during the development phase when the application has not yet been

The associate editor coordinating the review of this manuscript and approving it for publication was Jiafeng Xie.

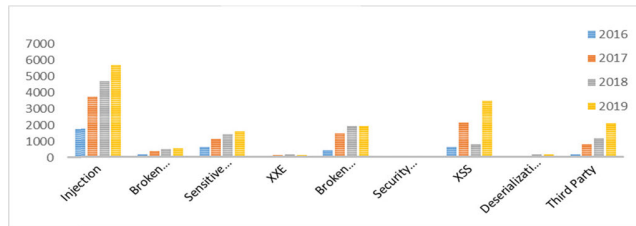


FIGURE 1. The state of web application vulnerabilities between 2016 and 2019 [3]¹.

deployed to a server. Static code analyzers aim to find code smells, bugs, and vulnerabilities for continuous code quality without the necessity of code execution. Examples of this group of vulnerability analysis tools are SonarQube [4] Parasoft [4], CodeSonar [4], Veracode Static Analysis [4], Fortify Static Code Analyser [4], Checkmarx [4], and AppScan [4]. In order to make automated static code analysis, the analyzer tool (SAST) which suits the development language and development framework has to be selected. Thus, to make a white-box test, the technology used in the development of the web application should be considered. Since, the number of development languages, platforms, and technologies is high, being dependent on the web application technologies and languages is a limitation for a security analyst.

The second group of vulnerability analysis tools focuses on black-box tests/analyses, and they do not depend on the selected technologies. They use standard HTTP requests to make controls and attacks on the web applications. They are more suitable to be used after the deployment of the application either to the test servers or to the production environment. These tools are commonly referred to as Dynamic Application Security Testing (DAST) tools. The tools in this group are also useful during the whole lifetime of the application and may help while making security-related design decisions after the first deployment.

In the security visualization domain, security-related data, which are log files, network traffic data, operating system data, data from security protection systems, such as firewalls, IDS systems, or vulnerability scanners, are visualized to provide more efficient and effective ways of security analyses. The focus of this study is to visualize the outputs of the second group of automated security analysis tools, i.e., DAST tools, which are typically the scan results, and the identified alerts.

Both SAST and DAST tools have their specific benefits for specific phases in the software lifecycle. Focusing on the DAST in this work should not imply anything related to the effectiveness of these two groups of tools. DAST tools have been selected, since more issues on the reporting features of these tools have been observed than SAST tools.

Web application security black-box test tools are called vulnerability scanner tools, in general. The types of vulnerability scanner tools vary. Port scanner tools, web server scanner tools, and web application scanner tools are the most

well-known types. The scans may be host-based or network-based. There are both commercial and open-source alternatives. Notable vulnerability scanners tools include Acunetix [5], Netsparker [6], Retina [7], Whitehat Sentinel [7], Burp Suite [8], Grendel Scan [7], Grabber [7], Nikto [7], and Zed Attack Proxy [9] (ZAP).

Bingham *et al.* [10] focused on the usability problems of black-box vulnerability scanner tools and attempted to identify the common pitfalls of user-interface designs, user notifications, and software configuration, based on two different tools. There are also a few general-purpose guidelines for usable security, such as [11] and [12] in the literature. Apart from earlier proposals and findings, the problems detected by the authors are as follows. Although these vulnerability scanner tools have reporting systems, these reporting systems are not adequate and mature to monitor the vulnerability status of software projects properly. When there is more than one web project to be monitored, then usability problems increase. More information related to the existing reporting and visualization capabilities of the vulnerability scanner tools is provided in the next section.

The primary contribution of this paper is a new dashboard tool for visualizing vulnerability scan results coming from black-box vulnerability scan tools. To achieve this, data attributes of these tools have been examined and a data structure has been formed including the attributes commonly found in the vulnerability scan results. A secondary contribution of this work is the list of metrics/measures that the tool presents (Table 1 to Table 8). The paper also describes a validation study in which the participants answer a user quiz on using the tool prototype. The work presented in this paper was carried out as part of the corresponding author's PhD study [13].

The proposed solution allows for dynamically inspecting and comparing the characteristics of vulnerabilities on multiple software projects or different versions of the same software project which are analyzed at different times. It aims to allow having a quick understanding of vulnerability levels, types, the association of these vulnerabilities to the standards, and the efforts and trends in security-related development, and security-related bug fixes for software projects.

The rest of the paper is organized as follows. The related work is described in Section II, followed by the process description in Section III. Detailed explanation of the holistic web application security vulnerabilities visualization approach and the metrics are provided in Section IV. In Section V, the case study that was developed using the proposed model including the validation efforts is demonstrated. The paper continues with the discussion of theoretical and managerial implications and research directions in Sections VI and VII. Concluding remarks are given in Section VIII.

II. RELATED WORK

Özdemir Sönmez and Günel made a detailed gap analysis for the security visualization domain [14]. This survey study included a detailed investigation of around 80 security

¹This figure includes only nine vulnerabilities instead of ten, because only nine vulnerabilities overlapped in the OWASP top ten lists between 2016 and 2019.

TABLE 1. Available metrics from well-known web application vulnerability scanner tools.

Vulnerability Scanner	Available Metrics
Acunetix	<ul style="list-style-type: none"> Web Site Based Aggregated Threat Level (low-medium-high) Total number of alerts for each category low, medium, high, informational Number of scan requests URL based result (ok, not found, moved permantly..) Scan time Average response time
Netsparker	<ul style="list-style-type: none"> Number of alerts for each severity category (information,low, medum, important, critical) Current scan speed (req/sec) Average Scan Speed (req/sec) Total Requests Total Failed Requests Number of Issues by Vulnerability Types Number of Issues by URLs Number of issues by severity Number of issues by confirmation Vulnerabilities by OWASP Top Ten Total Scan Duration
Retina	<ul style="list-style-type: none"> Number of web forms, Number of page with comments Number of pages with applets Number of emails Number of cookies Number of vulnerabilities by severity (safe, informational, low, medium, high) Number of vulnerabilities by vulnerability type for each severity group Total number of links Vulnerabilities by who will fix (Application developer, Server administrator, database administrator) Number of vulnerabilities by vulnerability types URL based vulnerability OWASPTop Ten, CWE, CAPEC and DISSA category
Nikto	<ul style="list-style-type: none"> Number of requests Total duration (start time and end time) Number of items (warning- alert) Number of errors
White Hat Sentinel	<ul style="list-style-type: none"> Pages tested in last completed scan Pages tested in current scan Stores scan information for more than one projects Mark as open/close Provides detailed information related to vulnerability such as attack vector and retestability status Number of vulnerabilities by severity Number of vulnerabilities by class Number of vulnerabilities by status(open/close)
Burp Suite (Free Edition)	<ul style="list-style-type: none"> Number of requests Number of issues Cross relations of vulnerabilities based on severity and confidence Number of issues based on severity Number of issues based on classes

visualization studies that were selected among about one thousand earlier studies from the Web of Science database. This extended review did not have a focus on a specific use-case or on a specific data source. On the other hand, it aimed to provide a clear summary of the security visualization domain for new beginners and for scientists who

seek gaps in the field. It contains classifications based on use-cases, data sources, interactivity properties, and display type selections. This survey study shows that although other security sources, such as network traffic data, have been given more importance, web application vulnerabilities still need more attention. This gap analysis resulted in the iden-

TABLE 2. Base measures/metrics based on vulnerability scanner tools.

Given Name	Meaning	Measure/Metric	Prototype Figure Num.
1 SetAlerts(t_n)	Alerts/vulnerabilities at $t=t_n$, {a1, a2, ax}	Measure	Fig. 7, Fig. 8
2 SetURLsProcessed(t_n)	{ url1, url 2, ..urlp} ²	Measure	Fig. 7
3 SetURLsScanned(t_n)	{ url1, url 2, ..urls} ²	Measure	Fig. 7
4 SetURLsWithAlert(t_n)	{ url1, url 2, ..urla} ²	Measure	Fig. 8
5 SetofNonAlertedsxPages(t_n)	Set of pages with no alert, URLsScanned(t_n)- ⁴ URLsAlerted(t_n)	Metric	Fig. 8
6 #NumVT(t_n)	# of alerts/vulnerabilities at $t=t_n$, SetAlerts(t_n) ¹	Metric	Fig. 9
7 #NumURLsProcessed(t_n)	# of URLs processed at $t=t_n$, SetURLsProcessed(t_n)	Metric	Fig. 7
8 #NumURLsScanned(t_n)	# of URLs scanned at $t=t_n$, SetURLsScanned(t_n) ¹	Metric	Fig. 7
9 #NumURLsWithAlert(t_n)	# of URLsWithAlert(t_n)	Metric	Fig. 9
10 #NumDistinctURLsWithAlert(t_n)	SetURLsWithAlert(t_n) ¹	Metric	Fig. 9
11 #NumDistinctURLsWithoutParamsWithAlert(t_n)	SetURLsWithAlert(t_n , trimmed,) ¹	Metric	Fig. 9
12 #NumPagesWithRptdAlerts	#of Repeated Alerts, SetURLsWithAlert(t_n) \cap^3 SetURLsWithAlert(t_{n-1}) ¹	Metric	Fig. 11
13 #NumPagesWithFixedVulnerabilities	#Pages whose vulnerabilities are fixed between phases, SetURLsWithAlert(t_{n-1}) - ⁴ SetURLsWithAlert(t_n)	Metric	Fig. 11
14 #/PerPageVulnerabilityLevelChange	Change in Vulnerability Level, (NumVT(t_n) - NumVT(t_{n-1})) / #NumURLsProcessed(t_n)	Metric	-*
15 #NumNonProcessedPages(t_n)	# of NonProcessedPages(t_n), URLsScanned(t_n) - URLsProcessed(t_n)	Metric	Fig. 7
16 %PctScannedPages(t_n)	Percentage of ScannedPages(t_n), URLsProcessed(t_n)*100/ URLsScanned (t_n)	Metric	Fig. 7
17 %PctAlertedPages(t_n)	Percentage of SetAlertedPages(t_n) over all scanned pages, #NumURLsAlerted (t_n)*100/ #NumURLsScanned(t_n)	Metric	-*
18 RTT	Round Trip Time for a scanned page	Measure	-
19 TotalRTT(t_n)	Total RTT in One Scan, $\sum_{i=0}^{URLsScanned(tn)} RTT$	Metric	Fig. 7
20 AvgRTT	Average RTT for a page, TotalRTT(t_n)/URLsScanned(t_n)	Metric	Fig. 7

- Not included in prototype
 * Can be calculated based on available information
¹ set elements count
² set parenthesis
³ set intersection
⁴ set minus

TABLE 3. Metrics-Measures/new developments-bug fix maintenance effect.

Given Name	Meaning	Measure/Metric	Prototype Figure Num.
21 SetWebPagesFixedDueBugFix (t_n)	URLs Fixed Due to Bug Fix $t=t_n$, { a1, a2, ..ap} ²	Measure	-*
22 SetPagesFixedDueToNewDevSet (t_n)	URLs Fixed Due to New Development $t=t_n$, { a1, a2, ..ad} ²	Measure	-*
23 SumTFixVul(t_n)	Total time to Fix Vulnerabilities between $t=t_n$ and $t=t_{n+1}$, $\sum_{i=t=t_n}^{t=t_{n+1}} securityVulnerabilityFixTime$	Metric	Fig. 11
24 #NumSecurityBugFix (t_n-t_{n-1})	# of security related bugs fixed in time period, SetVFixedDueBugFix (t_n) ¹	Metric	Fig. 11
25 #NumSecurityNewDev(t_n-t_{n-1})	# of security related new developments in time period, SetVFixedDueBugFix (t_n) ¹	Metric	Fig. 11

- Not included in prototype
 * Can be calculated based on available information
¹ set elements count
² set paranthesis

tification of several security visualization research topics. Visualization of web application vulnerabilities was one of them, which is the main motivation behind this work. The current study aims to provide ways to visualize web appli-

cation security vulnerabilities from DAST tools. For this purpose, another literature review has been made, focusing on web application security vulnerabilities this time. Web application vulnerability scanner devices have also been

TABLE 4. Metrics/Measures based on effects of previous measurements and time.

Given Name	Meaning	Measure/Metric	Prototype Figure Num.
26 AvgRL	Average Remediation Latency for a Vulnerability	Measure	-@

- Not included in prototype

@ The data collection is left as future work

TABLE 5. Metrics-Measures based on the effect of application properties.

Given Name	Meaning	Measure/Metric	Prototype Figure Num.
27 #NumLOC(t_n)	# of code length at time t_n	Measure	Fig. 6
28 #NumMdl(t_n)	# of modules at time t_n	Measure	Fig. 6
29 #NumExtLib(t_n)	# of external libraries at time t_n	Measure	Fig. 6
30 #NumWPT(t_n)	# total number of web pages in web app	Measure	.*
31 #/PerPageVT(t_n)	# of vulnerabilities per page at time t_n , #NumVT(t_n)/ #NumWPT(t_n)	Metric	.*
32 #/PerPageLOC(t_n)	# of vulnerabilities per line of codes at time t_n , #NumVT(t_n)/ #NumLOC(t_n)	Metric	Fig. 6
33 #/PerPageMdl(t_n)	#vulnerabilities per modules at time t_n , #NumVT(t_n)/ #NumMdl(t_n)	Metric	Fig. 6
34 #/PerPageExtLib(t_n)	#vulnerabilities per external libraries at time t_n , #NumVT(t_n)/ #NumExtLib(t_n)	Metric	Fig. 6
35 %PctURLsScanned(t_n)	Percentage of scanned pages, #NumURLsScanned(t_n)*100/ #NumWPT(t_n)	Metric	Fig. 7
36 ServerLocation	Server location for each application	Measure	Fig. 6
37 BaseURL	Base URL of web app.	Measure	Fig. 6
38 #NumBaseURLAlerts(t_n)	# of alerts for base URL at time t_n , NumVT(t_n) and url = BaseURL	Metric	.*

- Not included in prototype

* Can be calculated based on available information

TABLE 6. Metrics based on classification effect.

Given Name	Meaning	Measure/Metric	Prototype Figure Num.
39 #NumVLSIL(t_n)	# of Vulnerabilities/Alerts Related to Immediate Sensitive Information Lost	Metric	-@
40 #NumVLHBI(t_n)	# of Vulnerabilities/Alerts Related to High Business Impact	Metric	-@
41 #NumVLVC(t_n)	# of Vulnerabilities/Alerts Related to Vulnerable Components	Metric	-@
42 #NumVLH(t_n)	# of High Vulnerabilities/Alerts	Metric	Fig. 7
43 #NumVLM(t_n)	# of Medium Vulnerabilities/Alerts	Metric	Fig. 7
44 #NumVLL(t_n)	# of Low Vulnerabilities/Alerts	Metric	Fig. 7

- Not included in prototype

@ The data collection is left as future work

TABLE 7. Metrics based on the standards' lists effect.

Given Name	Meaning	Measure/Metric	Prototype Figure Num.
45 #NumVOWASP10(t_n)	# of vulnerabilities related to OWASP top ten list at time t_n	Metric	Fig. 13
46 #NumVWASC25(t_n)	# of vulnerabilities related to WASC top twenty-five list at time t_n	Metric	Fig. 13
47 #NumVCWE(t_n)	# of vulnerabilities related to CWE list at time t_n	Metric	Fig. 13
48 SetRLOWASP	Set of rules covering the CWE standard	Metric	Fig. 12
49 SetRLWASC	Set of rules covering the WASC standard	Metric	Fig. 12
50 SetRLCWE	Set of rules covering the CWE standard	Metric	Fig. 12

examined for their visualization capabilities. The details of existing web application security related visualization studies based on the review and examination results are provided below.

Dang and Dang [15] proposed a web application security vulnerabilities design with the aim of improving the reporting interfaces of vulnerability scanners. Their design was prepared to be utilized by security evaluators of web

TABLE 8. Metrics based on the protection systems' effect

	Given Name	Meaning	Measure/Metric	Prototype Figure Num.
51	#NumSecSys(t_n)	# of vulnerabilities prevented/blocked by security protection systems or other layers	Metric	Fig. 10
52	%PctVTSecSys(t_n)	Percentage of scanned vulnerabilities to detected ones, $\text{SecSys}(t_n) * 100 / \text{VT}(t_n)$	Metric	Fig. 10

applications. Dang and Dang's study aggregates data from multiple scanners and provides statistical information for each web page URL, based on the alerts gathered from these scanner tools.

Since the Dang and Dang's study forms the single example on visualizing the black-box vulnerability scan results of web applications, the following visualization studies were also included in this section to provide a better understanding of web application software security visualization concept:

- Security visualization studies, which aim visualization of static code analysis tools' (white-box analysis) vulnerability results for software (not necessarily web) applications,
- Security visualization studies, which do not use vulnerability data, but use other related security data, such as application security logs,
- Security visualization studies, which do not use security data, but provide helper information to be used during the tasks of web application security analyzers.

Cesar is a prototype proposed by Assai *et al.* [16] that aims to promote the usability of static code analysis tools by increasing the collaboration among software developers. Static code analysis tools are not specific for web applications, but all kinds of software. The authors of Cesar claim that contemporary static code analysis tools do not provide enough collaboration and this results in the software developers' unwillingness to use them. In our opinion, collaboration is not a feature that should be primarily expected from a vulnerability scanner tool. Definitely, any property which would enhance collaboration would be valuable for any tool. However, when we look at the software development domain, we see that there are already many tools that provide collaboration, such as task management tools (e.g. JIRA). Best practice development environments integrate the vulnerability scanners to be a part of a more collaborative tool to support the continuous development and integration processes. In this way, the outputs of vulnerability scanners can be shared among all the project team members continuously. Cesar visualizes the static code analysis results using the treemap technique. It also provides a way to jump to the application code from the detected vulnerability.

Other security visualization studies that deal with software related vulnerabilities are Goodall *et al.* [17] which visualizes software code vulnerability, and Harrison *et al.* [18] which visualizes the scanner results on NV, Nessus Vulnerability Visualization for the Web.

Security logs are important security data source alternatives for security visualization of web applications.

Alsaleh *et al.* [19] proposed a study that visualizes the security logs of PHP based web applications. This application aims to help security analysts during their examination of security logs. Attacks made to web applications and web-based attack scenarios are other visualization subjects related to web applications. In another work, Dang and Dang [20] proposed a system that visualizes web attack scenarios. This system depends on exploiting the links of web application pages and aims to understand intrusion detections. HVIZ [21] is a system that does not directly aim web application security, but it visualizes web browser activities. This design might be used for evidence gathering when an incident occurs.

Some visualization studies do not only use the security-related data, such as security logs, vulnerability scan results, or client access logs, but also provide some secondary information that eases the tasks of security analysts. For example, Dang and Dang [15] used website hierarchical structure along with website vulnerability scan results. The department data and user data areas were used as a part of Netvis [22] security visualization tool. These secondary data can be very helpful when used properly with the security data.

As mentioned in the previous section, there are known usability issues for developers using static code analysis tools [23]. This study aims to improve the reporting features of black-box web application vulnerability scanners, which are also not very convenient in terms of usability, for various reasons [10]. The number of abstraction levels is low for these tools. There is a need for more fine-grained abstraction. The difficulty of exporting scan results, the lack of definitions for vulnerability types and the lack of boundaries between different classes are other commonly reported usability issues. Some of these difficulties may be caused due to the difficulty of transferring information from the security domain to the software development domain.

The reporting systems of black-box vulnerability scanner tools, commonly serve the detailed scan results data and aggregated data based on alert types. They do not offer any other calculation or metrics, and they do not offer a way to make a transition between the aggregated data and the detailed data.

Considering that the focus of this paper is related to the presentation of data rather than scan data generation, a group of popular DAST tools has been examined in detail, to understand mainly their presentation features. DAST tools have a distributed range of features. Some of the tools allow you to choose groups of vulnerabilities to scan, such as in Acunetix [5], or they allow limiting the scanned domain

URLs by the use of some techniques, such as Regex mechanism as in Netsparker [6]. They provide ways to make several planned scans by allowing multiple scans queues, such as in Burp Suite [8], or scheduling scans for the future as in Acunetix [5]. In general, commercial DAST tools are more professional with a higher number of metrics and reports. Open source and free tools are simpler, mainly serving the scan data and the alerts. Some of the commercial tools also have community versions with fewer features.

The metrics/measures commonly presented by these tools are scan duration, number of requests, average response time, the number of locations, latest alerts, list of discovered hosts, list of vulnerabilities, including URL, type and parameters, details of the selected vulnerability, including the vulnerability description, attack details, and HTTP request detail (see Table 1). Standard metrics are the number of vulnerabilities by severity (such as high, medium, low, or informational) and the number of vulnerabilities by alert/vulnerability type as in Acunetix [5], which is one of the more advanced tools of this type. Netsparker [6] presents scan results/ scan logs, URL, scan duration, attack type, number of total requests, number of requests per second (speed), and average speed, number of failed requests, total time elapsed, head requests, alerts found and vulnerability description. Some tools such as Burp Suite [8] and Zed Attack Proxy (ZAP) [24] have simpler presentation styles with less number of metrics and measures. Burp Suite shows host, method, URL, params, status, length, MIME, title, IP, cookies, details of HTTP requests. In the alerts list (issue activity), status, issue type, URL, (host and path), and issue time are presented. ZAP [24] has similar output fields. These tools with simpler presentation designs also involve alert/vulnerability descriptions either short as in ZAP [24] or long as in Burp Suite [8].

Whereas the most of the data is presented through the tool screen views, few of the more advanced tools provide one or more type of reports, such as executive summary report as in Acunetix [5], developer, auditor, and administrator reports including OWASP top ten report, PCI compliance report, and knowledgebase report as in Netsparker [6].

Most of the web application scanner tools provide raw alert and scan data. A minimal number of metrics/measures related to the comparison of subsequent scans are included in the provided reports and screen views. Among the examined tools, it is seen that few tools allow manually marking the detected issues as “resolved”. This helps to track the status of security-related updates using the vulnerability scanner interfaces to some extent. Automatic comparison of multiple scans together with the integration of application and project data may provide better monitoring of security updates while examining current security issues for the web projects.

Although coloring and small icons for different levels of vulnerabilities are generally used both at vulnerability screen views and in the generated reports, nearly no other visual element exists at the output of the web application vulnerability scanners. Only a few of the tools allow filtering of the output data, as in the drill-down feature of the Acunetix [5], which

allows filtering based on severity, target, business criticality, status, and CVSS values.

The low number of studies focusing on web application vulnerability visualization, the usability problems of vulnerability scanner tools, and their immature presentation styles provide challenges to the researchers. These challenges include definition or identification of new metrics and providing new visualization designs that will enable monitoring of these newly identified metrics, changes between multiple security checks, and effects of new developments and bug fixes.

Dang and Dang’s study which is a single example of visualization focusing on DAST results mainly presents statistical metrics. It provides the total numbers of vulnerabilities each scanner found and further divide these numbers into groups by common severity levels of vulnerabilities. It uses web application pages’ structure during this presentation. Although this approach provides a very top-level view of the vulnerabilities, it lacks details. Besides, this view does not show repeated alerts, fixed alerts between phases, and it does not relate the alert data to the standards. Software project managers’ responsibilities include reviewing the current status and progress against intermediate and final development targets and identifying the obstacles. Managers should also monitor whether the security implementations meet the standards and technical requirements. Including repeated alerts, and fixed alerts between the periods may be valuable for the managers besides security analysts, and would end up with a new visualization perspective for the same type of data.

Existing web application vulnerability visualization tools do not have a holistic approach that combines vulnerability scanner results with the environment properties and timeline of security-related activities. They also lack clear metric definitions and the number of metrics they present is very low in general, resulting in incomplete picture of the security status of web applications. Security Information and Event Management (SIEM) tools, on the other hand, provide a more holistic approach, with a large number of metrics for the enterprises. However, rather than focusing on web application security, they aim to serve other purposes, such as reporting and managing security alerts in real time based on the analysis of data obtained from the entire IT infrastructure.

Against this background, the purpose of this study is to propose an alternative visualization tool which visualizes the data attributes commonly available for web applications, combine these data attributes with common outputs of web application vulnerability scan results, namely, scanner results and alerts, and to find out measures and metrics based on the proposed data structure. For this purpose, in parallel to designing the metrics and visualizations, existing SIEM systems are also examined in detail to discover their potential benefits for the monitoring of DAST results.

III. PROCESS DESCRIPTION

The top-level processes included a literature review of the academic studies. This was followed by a detailed

examination of the existing vulnerability scanner tools, which resulted in understanding the capacities and problems of these tools in depth. As an initial step, a data structure was framed as the outcome of analyzing typical results of vulnerability scanners and selected secondary data sources.

An essential part of this study is defining quantitative metrics. Following the data source structure formation, a large set of metrics/measures were identified using the proposed data structure. Encapsulating a large set of metrics is not very common in security visualization solutions. Similarly, explicitly marking each metric in security visualization solutions is also not accomplished yet. Although statistical results are distinctly highlighted in the security visualization solutions, other metrics are left to the users' understanding.

Prototyping is the most widely taken approach in the security visualization domain to illustrate the novel visualization designs. For this purpose, to illustrate the proposed approach, a visualization prototype tool was developed in dashboard form, called Holistic Web Application Security Vulnerability Visualization (HWAS-V). The dashboard-style was selected due to its ability to encapsulate a large number of metrics in one design and using a limited space easily. During the preparation of the dashboard, several combinations of the metrics were built experimentally, and among them, the most appropriate combinations were selected to form a legitimate dashboard design. More information is provided related to the design decisions and rationalities in the following sections.

As part of the design description, the details of the metric definition process and output metrics are described in the next section. The primary and secondary data sources are also explained in this section. Later, the offered metrics are depicted in a classified manner. Next, the visualization process is revealed in detail. Although a detailed examination of existing web application vulnerability scanner tools was conducted in the previous section, evaluation of SIEM systems is also included to position and differentiate the proposed system among the SIEM systems. A short summary of the evaluation results of the SIEM systems for their potential benefits to web application visualization is provided as part of the tool description in the next section.

A case study was designed to demonstrate the usage and benefits of the HWAS-V prototype. For this purpose, case-study vulnerability data was formed by conducting subsequent vulnerability scan analysis for three different domains. This data was used during the visualization of the provided metrics.

The authors had sought appropriate ways to validate the provided system containing proposed metrics and visualizations. Unfortunately, a suitable criteria to validate the security metrics were not encountered in the literature. The closest match was the software metrics validation criteria, which were obtained by examining a spectrum of philosophies in the study by Meneely *et al.* [25]. The criteria provided by Meneely *et al.* are valuable, consisting of every aspect of validating a metric, however, it is based on a list of all 47 validation criteria including "Empirical validity", "Monotonic-

ity", "Instrument validity", "Representation condition", "Actionability" and numerous others. Making that kind of evaluation for a single new metric or a few new metrics may be feasible, but not so for this study having about 50 metrics. Another issue is although these evaluation criteria were very suitable for the software domain, not all the evaluation criteria are equally suited to the security metrics. Some of the evaluation criteria are very philosophical indeed, very complicated for the technical evaluators.

For the listed reasons, rather than validating the metrics and visualizations one by one, the authors decided to enable validation of the whole system based on selected criteria that suit the aims of this study. For this purpose, in the last part of this study, a validation survey was administered. During this validation survey, the survey attendants were given the data and the visualization prototype, and then they were asked questions. This validation study included 15 quiz questions which should be answered using the HWAS-V prototype. The quiz questions were created to allow the users to use every part of the HWAS-V tool while trying not to exaggerate the number of questions. After this quiz, a set of four questions were asked the participants to measure the practicality, efficiency, decision-informing, and difference-detection attributes of the proposed system. These attributes of HWAS-V were questioned using a five-point Likert scale mechanism, from 1 to 5 meaning 'Not Helpful', 'Slightly Helpful', 'Helpful', and 'Very Helpful'.

IV. HOLISTIC WEB APPLICATION SECURITY VULNERABILITY VISUALIZATION, HWAS-V

In general, the users of web application vulnerability scanners are expected to have technical competency having positions, such as security analysts, and system admins. To provide a system that gives a broader perspective of security statuses, a new approach is introduced in this study. In this approach, the project-level details, such as earlier security analysis results, application-level details, such as the size of the application, the number of modules, the number of external libraries, and the standards related information are integrated with the vulnerability scan results. The proposed security visualization solution is expected to be valuable for various roles including a software project manager and a security analyst. The efficient usage scenarios of this solution for six roles are demonstrated in Section VI.

A. METRIC DEFINITION PROCESS

Vulnerability scanners may have a variety of different focuses; still they have similar working mechanisms. These tools make scans based on defined rules, "scanner rule". During the scans, they generate the "scan data" which include requests and responses and the resulting "alert"s. It is difficult to form a generic data structure which supports all web application scanners, because each scanner would have its own attributes and data types. Thus, during the design of the data structure, the selection of mandatory data attributes was made by using a minimum set that commonly

exists in the web application vulnerability scanner results. There are also some optional data attributes. These attributes take part in relating the minimum data set to some other data. For example, `CWE_id` in alerts and `scan_rules` relate alerts and scan rules to some existing standards information. The prototype is designed so that, if there is no association information for the selected automated vulnerability scanner tool, this does not affect the overall visualization system. If there is a known relationship, this relationship is used by the visualization system to provide some additional metrics.

The scan rules defined in a scan tool is the primary data source used for visualization purposes in this study. These rule definitions may include a “rule name”, and a “rule id”. There may be some additional categorical measures, such as “version” information. Optionally, there may be information, which relates “rule” to the security standards. Although this type of data does not change frequently, it provides information related to the coverage efficiency (scope) of the vulnerability scans, and be used to relate scan outputs to the common standards, such as Open Web Application Security Project (OWASP), Common Weakness Enumeration (CWE), and the Web Application Security Consortium (WASC).

At the beginning of a vulnerability scanning activity, a base URL is needed. Once the base URL is identified, automated scanner tools check for all available pages in that domain, thus, form a list of all available pages for that web application. Forming such a list is called spidering or crawling in the web terminology. The result of spidering is called “scan results” in this study. These results include information related to scanning of the base URL and related pages which are found during the crawling. The scan results may include information such as “scan id”, “process result”, “request timestamp”, “method”, “URL”, “response code”, “reason”, “RTT”, “request header size”, request body size”, “response header size”, “response body size”, “highest alert”, and “tags”. Tags are optional, and they may not be available for all scanner types, but similar to the relationship to standards, existence of this information may provide some additional metrics. The second primary data source of this study is the “scan results”.

Once a URL is pointed out to the vulnerability scanner tool, and all the pages are crawled, the tool filters the pages that do not belong to the target domain. Later, it checks the results of applying the scanner rule for each page. The checking mechanism may depend on passive controls or active attacks. In the end, it provides a list of alerts associated with a list of instances where each instance corresponds to a URL. These results include the “alert name”s, “URL”s of the related pages. Alert names may be equal or similar to the corresponding “scan rule name.” Unfortunately, there is no standard for naming the scan rules and the corresponding alerts for vulnerability scanner tools. Mapping of scan rules to alerts can be made for once by the tool users to be benefited during

the subsequent scans. The alerts list is the third primary data source of this study.

To form the measures/metrics list, earlier academic or commercial published material that points to the web application security metrics were examined. Later, this initial set of metrics were extended by including the measures and metrics selected from the ones that can be generated using the proposed data structure.

Thus, a few of the proposed metrics were mainly designed, because they were convenient to measure using the available data. Considering the aim is not to find a solution for all types of problems of security analysts, but to improve the ways of examining the web application vulnerability data, this was an adequate approach. Knowing this, and the examinations’ results, which indicate the low number of measures/metrics and lacking abstractions in multiple levels, the proposed metrics and measures were most appropriate for the security analysts and other potential users providing various abstractions of vulnerability data in different levels.

Before the definition of the proposed measures and metrics, it is necessary to explain the terms, “project/domain” and “phase” used in this study. “Project/domain” corresponds to either a web application that is in the development stage and subject to the security tests (i.e., a project), or an already deployed application which is the subject of a security analysis/test task (i.e., a domain). Since such security analyses/tests are repeated actions, a “phase” definition was required to measure the effects of repeated vulnerability scanner results and the changes made after each security analysis. Hence, whenever a new automated test is recorded for a project/domain, a new phase starts. These phases do not necessarily correspond to the phases in a project life cycle. This phase concept is necessary to enable measures related to changes due to vulnerability fixes and bug fixes or new developments in a defined phase in the proposed framework.

Fig. 2 shows a series of vulnerability scan results achieved for two independent projects. In all lifetime of the project 1, the vulnerability scanner was executed five times. For project 2, the vulnerability scanner was executed only in the maintenance phase, four different times.

During the metric design phase, application properties, such as application size, the number of modules, and the number of external libraries are used in conjunction with vulnerability scan results. If the development continues for the application, then this information will change from phase to phase. Data coming from security protection systems are another type of secondary measures used in the proposed framework. These measures are combined with security standards-related information to form a holistic data structure which enables monitoring security statuses of the web applications for multi-project, multi-phase platform combined with an association to the standards and other information related to the project structure and the environment. In summary, measures related to the application properties,

text/analysis phases, security protection systems, and standards are included as secondary data sources in this study. The implementation related details for this tool is presented in the Appendix Section. Appendix A includes a conceptual system model that might be used to collect the data for the proposed data structure. Appendix B shows the proposed data structure. Using data from multiple sources simultaneously has normalization problems inherently. In this study, there are time-independent data, such as standards data, and remediation data, and time-dependent data, such as vulnerability scan results, application data, and project data. Having the phases becomes advantageous when solving the normalization problems. When time-dependent data from different sources, such as the project management tool or the vulnerability analysis tool, are pulled, accessing a normalized data would be possible if the corresponding phase's start and end dates are adhered to in both manual and automatic data loading options.

When there are no phases, i.e., predefined start and end dates for each vulnerability scan period, capturing the changes in the application modules, the number of vulnerabilities and the project tasks would occur at different frequencies and different calendar dates. However, when there are predefined phase start and end dates, the user of the system would be forced to make specific checks for the stored attributes (such as the application sizes, number of security-related tasks, and vulnerability scans for these predefined intervals) at specific times. Therefore, even if the lengths of the periods and the frequencies of security checks may differ for a project, the system would provide comparable values using all the measured attributes for each phase in itself. This reliance on phases should be used for both manual data entrance/upload and automatic queries from the integrated tools.

For a project, time dependent and time independent data co-occur. For example, there may be tasks and changes in the application state, which are time dependent. On the other hand, the vulnerability data is like snapshot data. In order to handle such time dependency, snapshot data should be marked with the same time intervals as the time dependent data, so that they can be served in the same dashboard to have a holistic view of the security statuses.

The proposed measures/metrics are the results of primary vulnerability scan results and related factors, such as the information related to new developments, bug fixes, maintenance effect, time effect, classifications, application properties, protection systems, and the standards. The metric/measures are presented using this classification for better understandability.

Although a generic data structure is proposed in this paper, the examinations of other security test tools (presented in the related work section) showed that we do not leave out any important attributes which are specifically used to characterize some security issues. Even if the tool interface and available reports may differ, the DAST tools work in the same manner, in general, through HTTP calls, and thus provide similar outputs.

B. WEB APPLICATION VISUALIZATION MEASURES/METRICS BASED ON COMMON VULNERABILITY SCAN OUTPUTS AND RELATED DATA

Measure and metric are two terms, which are sometimes used interchangeably in some contexts. The main difference between them is that the measure is the direct result of a measurement activity; metric, on the other hand, is the result of a calculation made using one or more measures. In this study, these terms are used in compliance with the provided definitions. The measures and metrics that are available using the previously described data structure are explained in detail in this section.

1) BASE MEASURES/METRICS BASED ON VULNERABILITY SCANNER TOOLS

The measures in this group are originated from the web application security vulnerability scanner tools. In this study, these measures are called base measures, because they are based on measurements from the primary data source. In the subsequent sections, this base list is enlarged by the potential effects of secondary data sources. The measures alert set, set of URLs scanned through vulnerability scanner, set of URLs processed, set of URLs associated with an alert, round trip time (RTT) for an HTTP request and response for each alert check provided by the scanner tools are among the base list. The metrics the number of vulnerabilities/alerts, the number of URLs scanned, the number of URLs processed, the number of URLs with an alert, the number of repeated alerts based on alert data, the number of fixed alerts based on alert data, change in vulnerability level per page, the number of non-processed pages, the number of non-alerted pages, percentage of processed pages, percent-age of alerted pages, total RTT, and average RTT are the results of using simple arithmetical or set operations on the measures listed in this group. The associations of the metrics/measures in this group to the dashboard parts are shown in Table 2.

These measures/metrics serve various purposes including providing raw alert/scan information through sets, demonstration of the level of application/project security, the level of scan coverage success through basic arithmetic operations, providing the level of success of vulnerability/alert fixing efforts, such as the number of repeated alerts, the number of fixed alerts, and the change in the level of vulnerability levels through set operations, and presenting the scan duration related information which may be used for the planning of subsequent scans and to have an understanding of the performance of the applications/projects.

2) METRICS DUE TO THE EFFECTS OF NEW DEVELOPMENTS / BUG FIXING MAINTENANCE

During the lifecycle of a web application, both new developments and bug fixes might exist in time. In order to monitor the effects of these tasks, these efforts might be associated with the previous findings of the vulnerability scans. Once

these associations are made, it is possible to include the following measures and metrics to the proposed system.

The measures due to the set of alerts/vulnerabilities fixed, bug fixing related tasks, the set of alerts/vulnerabilities fixed due to new developments, and the total time to fix vulnerabilities between the scans can be measured through the use of task management systems properly. Arithmetic operations calculate the metrics, the number of related bugs fixed in a period and the number of security-related new developments. These metrics are used in conjunction with the metrics presented in the base measures/metrics section.

The metrics and measures proposed in this part would form an internally developed remediation latency database for the known type of alerts for that specific type of application in the long term and may be used for planning purposes. The associations of the metrics/measures in this group to the dashboard parts are shown in Table 3.

3) METRICS/MEASURES BASED ON THE EFFECTS OF PREVIOUS MEASUREMENTS AND TIME

When time passes, the system/software would eventually undergo some changes. The effects of new developments and bug fixes were already mentioned in the previous section. In this part, the internal and external measurements, namely the remediation latency indicator values collected by IT companies in time for common alert types are included in the proposed metric/measure list. Remediation latency is an indicator that would measure the security update performance of the developer organization. IT companies periodically announce such information related to common security issues. Integrating such information with the vulnerability scanner results would be beneficial for multiple purposes such as planning of new developments/bug fixes and monitoring of the progress of the ongoing projects. Average remediation latency for a vulnerability is either a measure gathered from vendor report or a measure that is ascertained during the tasks described in Section B.2. The list of metrics/measures in this group is shown in Table 4.

4) METRICS/MEASURES BASED ON THE EFFECT OF APPLICATION PROPERTIES

The size of the application, the use of internal or external libraries, and integrations made with third-party tools would affect the security status of the web application. The measures and metrics related to this information are also integrated with vulnerability scanner results to provide a more holistic view of the web application security.

The measures related to the application properties used in this study are the elements which indicate application size in various ways, such as the number of lines of code (LOC), modules, and web pages at time t , and the elements which show information related to the application deployment structure, base URL and the geographic deployment location. The latter parameters are used for monitoring the application status in a map. The metrics designed by the authors using these measures are the number of alerts for

base URL, the percentage of scanned pages to total pages, the number of vulnerabilities per LOC, vulnerabilities per module, and vulnerabilities per web page. Alerts for the base URL is calculated through the use of string operations on the scan results. These are the alerts related to the base URL regardless of the extension of the URL with the alert. The percentage of the scanned pages is the fraction of the scanned pages over the total number of web pages provided in the application properties multiplied by 100. The associations of the metrics/measures in this group to the dashboard parts are shown in Table 5.

5) METRICS BASED ON THE CLASSIFICATION EFFECT

The majority of vulnerability scanner tools provide information that would help categorization of the alerts/vulnerabilities. The most common categorization criterion is the severity/importance level, such as high, medium, and low. There may be other categorization items, such as the existence of sensitive information risk (“sensitive information risk exists”, “sensitive information risk does not exist”), the effect to the business (“high business impact”, “low business impact”), and the origin of the vulnerability (“a vulnerable component exists”, “a vulnerable component does not exist; vulnerability is due to other issues”). Making these categorizations is commonly the responsibility of the scanner developers, due to the fact that they are the people who know the inner mechanisms and targets of the attacks/scans.

Besides the developer efforts, the standards information may also be benefited from while providing these offered vulnerability classifications. Although the scope of this work does not include making these classifications of the vulnerabilities/alerts, corresponding measures/metrics are included in this section, as shown in Table 6.

The number of vulnerabilities/alerts related to immediate sensitive information lost forms a group of vulnerabilities that are directly related to the loss of any sensitive information. Not all vulnerabilities result in information loss, so the vulnerabilities in this group should be directly associated with information loss. Similarly, the number of vulnerabilities/alerts related to high business impact would change from business to business and may be difficult to detect. The vulnerabilities of this group need not be related to information loss. For example, a DDOS attack that has a business impact, but no information loss would be in this category. The number of vulnerabilities/alerts related to vulnerable components can be identified during an examination of the application structure and its possible relations to the alerts. Some alerts may not be directly related to the application structure, but may be due to external effects. Such vulnerabilities are not counted in this group. The associations of the metrics/measures in this section to the dashboard parts are shown in Table 6.

6) METRICS BASED ON STANDARDS' LISTS EFFECT

The majority of the alerts have associations to available security standards, such as OWASP, WASC, and CWE. These association values may be used to calculate new metrics as

part of the web application security monitoring dashboard. Vulnerabilities/alerts covered from the OWASP, WASC, and CWE standards are found out using published associations of scan rules to the OWASP, WASC, and CWE standards. The associations of the metrics/measures in this group to the dashboard parts are shown in Table 7. The measures/metrics listed in Table 7 form two groups. The first group demonstrates the number of vulnerabilities for each selected standard, the second group demonstrates the coverage of those standards through the scan rules. Although some of the existing SIEM like security systems provide associations to one or multiple security standards, they do not provide information regarding the level of coverage of scan rules based on these standards. In general, scan rules implemented in the vulnerability scanners need not cover all parts of these standards. There are even some scanners that deal with one or two types of vulnerabilities, such as SQLInjection. Thus, without knowing the coverage level, knowing the number of vulnerabilities associated with the standards may not be likewise beneficial.

7) METRICS BASED ON PROTECTION SYSTEMS' EFFECT

In an environment where continuous monitoring of the web application security exists, it is commonplace that there might be other security protection systems. The measured number of vulnerabilities prevented/blocked by external security protection systems and the ratio of scanned vulnerabilities to detected ones are also included in the proposed web application security monitoring system. The number of vulnerabilities prevented/blocked by external protection systems is a numerical value that can be gathered from systems, such as firewalls, antivirus and antispam software, and IDS. The ratio of scanned vulnerabilities to the detected ones is the fraction of vulnerabilities prevented/blocked by external protection systems to the total number of vulnerabilities detected by scanner systems multiplied by 100. The associations of the metrics/measures in this group to the dashboard parts are shown in Table 8.

C. VISUALIZATION OF METRICS

The motivation for visualizing the proposed metrics resulted in a dashboard design that integrates automated vulnerability scanner results with other related data sources providing a summary of the vulnerabilities and their relations to the application, the system, and the environment. In this way, the design presents security-related highlights and eases the monitoring and tracking of security statuses of one or more projects in multiple phases. The outstanding features of the proposed system are supposed to be its practicality, its efficiency in analyzing the data, and its decision informing and difference detection capabilities. The existence of these expected features was tested explicitly during the validation part of this study.

The visualization of the metrics depends on several hierarchical decisions. Before going into the details, the rationalities of these decisions are provided as follows. The top-level decision was using a dashboard-type display. This was a

rational decision due to the large number of metrics. The second level decision was made when arranging the metrics into multiple dashboard views. This was again logical due to the number of metrics. As a part of this decision, the groupings of the metrics were made so that the final set of designs go from general to detailed and also present the needs of different roles in the most optimal way possible. Sample usage scenarios for different roles are visually described in the following sections. Although the responsibilities of roles and their tasks may vary from organization to organization, these sample scenarios show that each selected role can benefit from the proposed system by minimum interaction with the dashboards. These groupings provide visibility of the system status while matching the system with the real world. Decisions on a lower level are related to the general appearance of the dashboard views. These views have a standard and consistent structure [26], using plain titles, and subtitles, repeating font styles, and similar color schemes providing an appealing interface for the users. As a rule, the creation of an appealing visualization should never be the main goal for the designers. However, creating aesthetically good-looking designs affects the usability of the tools positively. Having an aesthetic and minimalist design is also one of the Nielsen's [26] ten usability heuristics rules.

Decisions on the lower level are related to lower-level chart selections. These charts are mainly selected among various types of simple 2-D charts. Using simple 2-D charts, such as bar charts and pie charts, makes it easier to comprehend and monitor multiple measures/metrics simultaneously. 2-D charts result in less false readings and enable better comparison of items. The overlapping of data points in 3-D charts commonly results in misunderstanding of patterns. As the number of measures/metrics increases, their interpretation via 3-D charts or other complex charts would become even more difficult.

The rationality of the selection of each specific chart relies on general visualization rules based on the data type and the use-case. For the comparisons of percentages or proportions, such as the percentage of pages processed or the pages successfully respondent, the pie chart is used. If the number of items to be compared does not allow the use of a pie chart, the sizes of markers are used to point out significant information, such as the URL pages with a higher number of alerts. A map is used to show location information. A bar chart is commonly used to show the frequency of simple and aggregated values. Treemap diagrams are used to visualize hierarchical structures. To avoid overwhelming the design with too many bar charts marker charts are used a few times. To prevent eye strain and confusion when markers are used in the same dashboard, different marker shapes are selected for multiple charts. In all parts of this design, color is used to differentiate categorical groups. In terms of color coherency, although all charts consist of somewhat related data, repetition of the same data occurs only in two dashboards to show application size information where the same colors are used in both charts.

Tableau desktop software [27] was used for the creation of the dashboard-based visualization prototype, due to its convenience and capability to create complicated dashboards. Tableau provides flexibility in the views by allowing drag and drops and resizing mechanisms. Having explicit filters in every dashboard, and due to Tableau's inherent filtering mechanism, which is activated during the navigation over the data automatically, allows the user control and freedom [26].

Looking at the conceptual part of the design, as explained along with the tool description, in this concept, the "project" refers to the actual web application which is either in development or production state. Sometimes the keyword "domain" is also used to refer to the "project" throughout the text. The phase refers to a duration that ends with a tool-based security analysis for a web project. Thus, the duration of the phases will be very variable. If frequent automated security analyses are done for a domain, then there will be more phases. If a new analysis is made and recorded, then a phase is finished. Thus, the analysis end time defines the phase end time, and phase start time is identified by the previous phase's end time or the project start time. As a result, the design enables analyzing repeated alerts, effects of bug fixes, new developments, and environmental changes between the phases. The "scanner" refers to a vulnerability scanner that provides data close to the proposed model specification, meaning "providing a list of scan data and alert data associated with URLs and alert types".

The aim of visualization design is to enable visualizing the automated vulnerability scan results as is. Thus, the vulnerability scan results are planned to be visualized without any cleansing or modification operation, for quick response, (see Fig. 3 (a)). The only data preparation step prior to the visualization and after data collection is due to the automatic generation of Id values. During the data preparation phase, in order to merge multiple data sources and differentiate subsequent executions of the automated scans some numerical Id values are generated. For example, "phase id" is generated to identify each execution of vulnerability scans and determine the duration passed between each automated test/analysis period. "Project id" is generated to be used in the creation of sets for the dynamic calculation of sets among projects and phases. Otherwise, dynamical text processing capabilities of the tool are utilized to make conversions, such as splitting united data, such as URL strings including the method (e.g. Get, Post) or elimination of parameter information from the URLs to determine the unique number of alerted pages.

Although the static data which was specifically collected for the use-case based on technical documents is used to make associations of other data parts to the "Standards" table, dynamic classifications are also made using set operations of the tool over the data. The reasons for using set theory were the necessity of grouping the alert data based on multiple fields and the necessity of using set operations such as union, intersection, and set minus. For example, to find "repeated alerts", and "new alerts" in different phases, set intersections and set minus operations are used over URL

datasets, respectively. These set definitions are depended on predefined project-id and phase-ids. When the numbers of projects or phases (repeated vulnerability scans) exceed the predefined maximum values, then these rotating Id's should be reset by the system, as shown in Fig. 3 (b).

New numerical values for some categorical values are created through calculated fields, such as the "numerical alert level" which is created from the categorical alert level attribute. Aggregation is often used for many purposes, such as aggregation of data based on scan rules, projects, and project phases. Besides aggregated data, the proposed visualization system also includes detailed data, such as page-level alert information.

Due to the high number of proposed measures/metrics, several dashboards are created which focus on different aspects of the data. These dashboards are aggregated in a "story" which is a feature of the tool that allows easy navigation among multiple dashboards. Some of the visualizations may be the results of some straightforward arithmetic calculations on a single data source. Results that are single numeric values are often visualized using "Formatted Text"s. Results which include a series of numeric values are visualized using charts, such as "Bar Chart", and "Pie Chart". Some other visualizations require joining or blending data from multiple data sources and may use other 2-D display types. Tables are used to show some detailed data, such as URL based detailed information. Bubble charts are used to show some grouping effects, such as grouping based on standards. The dashboards prepared as prototypes are presented through a case study.

Encapsulating a large set of metrics in a dashboard design requires using the space effectively. Frequently, a small portion of the view area has to be used to present metric values having wide ranges. From time to time, the results shown in a chart may have very close values, causing overlapping of the data points. On several charts, logarithmic scales are preferred to normal scales on data distribution on the axes to overcome these difficulties. A novel visualization property in the proposed dashboard design is the explicit association of the proposed metrics to the charts. Showing tooltips for charts automatically or on-demand is a feature of the design tool. These tooltips are used to show additional information, such as detail data, values of related attributes, etc. for all charts (see Fig. 3 (c)). Besides this additional information, this space is used to show the association of the display part to the named metric in a formatted and colored manner to improve the usability. This property will make it easier to understand and interpret the values for the users when navigating through the detailed dashboard designs. The use of additional textual explanations is not very common, but exists in the security visualization domain. For example, including human readable explanations of the patterns was included by Tri and Dang [28] to improve the understandability of the models.

The resulting visualization system consists of multiple dashboards that are fragmented based on the logical grouping of metrics and ordered based on some logical flow of

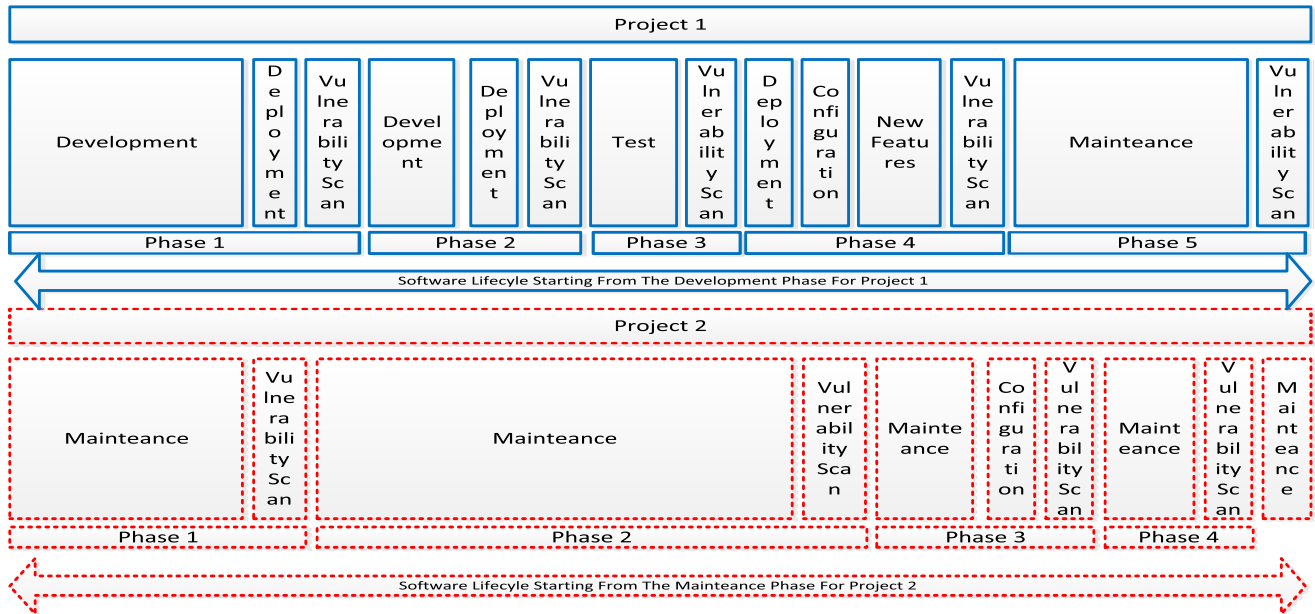


FIGURE 2. Multi-project multi-phase vulnerability scan results.

information. In Tableau, this demonstration form is called a “story”. The story allows navigation among multiple dashboards easily and allows an explicit description of each dashboard. The aggregation of these definitions indeed forms the story itself. Serving the metrics in a grouped manner based on several titles would help to have more information and to make a transition between aggregated data and the detailed data without getting lost in details, and reveals the relation of vulnerability data with other data parts in a clear manner. Using the tool, the users can examine the status of the security vulnerabilities and relationship of vulnerability data with other data sources in a systematic manner.

A few of the proposed metrics were excluded from the dashboard prototype based on multiple reasons. The first reason was the lack of corresponding data for the case study. For example, collecting average remediation latencies of known vulnerabilities was out of the scope of this study. Thus, not having the corresponding data, no visualization was included related to remediation latencies in the prototype. Another reason was the limited space available in the proposed dashboard. Although some charts and tables included such detailed information, some of the measures/metrics which include sets of URLs were not visualized in the prototype on purpose. Such detailed information may be served on-demand using other ways, such as tooltips connected to blank sheets which do not take much space in a real product.

D. EXAMINATION OF SIEM SYSTEMS FOR SIMILARITIES AND DIFFERENCES TO HWAS-V

To further investigate the similarities and differences between the SIEM systems and the proposed tool, six SIEM systems, Manage Engine Event Log Analyzer [29], Splunk [30],

Rapid7 InsightIDR [31], Solar Winds Log and Event Manager [32], Micro Focus ArcSight [33], AlienVault [34], which are located in four quadrants of the Gartner [35] analysis were installed on a test machine as an extension to this study. SIEM tools are more successful in working with continuous real-time data. The data structure proposed in this study is not for continuous data, but data is collected intermittently based on the project schedules including the maintenance phases.

The results indicate that SIEM tools do not have a specific focus on the web application vulnerability scan results. Although a few of them can integrate with some vulnerability scanners (mostly network vulnerability scanners), they do not provide a built-in data structure that will fit most of the web application scanner results. Prebuilt metrics specific to web application vulnerability scan results are very low compared to HWAS-V or do not exist at all. A few of the SIEM systems allow importing custom data, which may allow the creation of part of the visualizations presented in this study. However, SIEM tools do not have data joining, data blending, and set operation features comparable to HWAS-V, which relies on the Tableau business intelligence tool. The detailed information regarding evaluating results for custom visualization generation capabilities of SIEM systems and available metrics related to web application security domain can be found in Özdemir Sönmez and Günel’s work [36].

V. CASE STUDY AND HWAS-V

This part includes information related to the case-study data, case-study prototype, and the description of the evaluation study. The evaluation study consisted of using the prototype, answering the quiz questions and evaluation of answers afterwards. Data formation was the initial task of the case study.

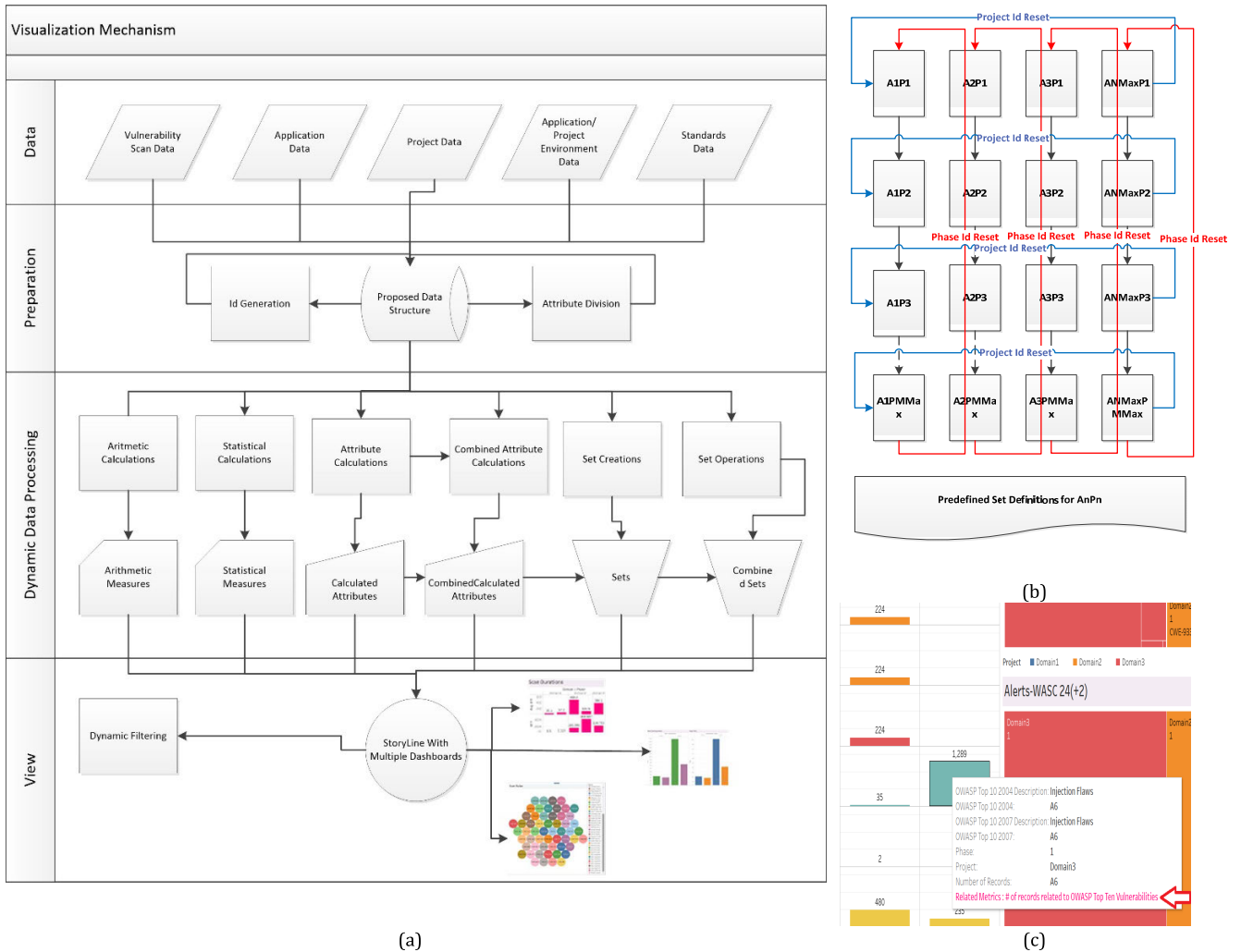


FIGURE 3. (a) Visualization mechanism top level components (b) Automatically generated IDs rotating to cover multi-project, multi-phase data (c) Explicit metric descriptions on the tooltips.

OWASP Zed Attack Proxy (ZAP) (OWASP, n.d.) automatic web application vulnerability scanner tool was used to generate the vulnerability scanner data for this case study. OWASP ZAP was selected, because it is a free tool and it has an easy to use interface. Compared to other commercial and non-commercial alternatives, it is widely used and positioned well in the spectrum of application security testing tools (#6 among 30 tools [37]). OWASP ZAP tool is a proxy application that combines a number of features including spider tool, active scan, passive scan, port scan, rest API, and the reporting functions. For each scan type, rules are defined by the community users (contributors), and independent evaluators evaluate these scan rules, prior to integration with the tool.

OWASP ZAP has various modes, standard mode, safe mode, protected mode, and attack mode. In this study, “attack mode” was selected, because this mode provides a higher level of information related to the targeted domains. Once a URL is pointed out; first, the tool crawls all the URLs in that domain. Afterwards, it filters the URLs which do not belong

to the target domain. Later, it makes attacks to the selected pages. Lastly, it provides a list of alerts associated with a list of instances where each instance corresponds to a URL.

OWASP ZAP attack tool was utilized on three independent domains several times to provide data related to the scanning results and the alerts. Later, this initial data was anonymized to some extent and combined with some data related to other aspects of the proposed visualization system to form a mockup dataset for demonstration purposes. The resulting mockup dataset includes all the data attributes shown in Fig. 3 (a) for three independent domains for two analysis phases.

A comparison having two groups of users such that one group uses OWASP ZAP raw output instead of HWAS-V and the other uses HWAS-V was not carried out. The efforts of the group having raw data would not be a quiz, it would turn into a data analysis study that would take hours, maybe days. Actually, deciding the time that should be given to the group which would be using raw data would be problematic. If a

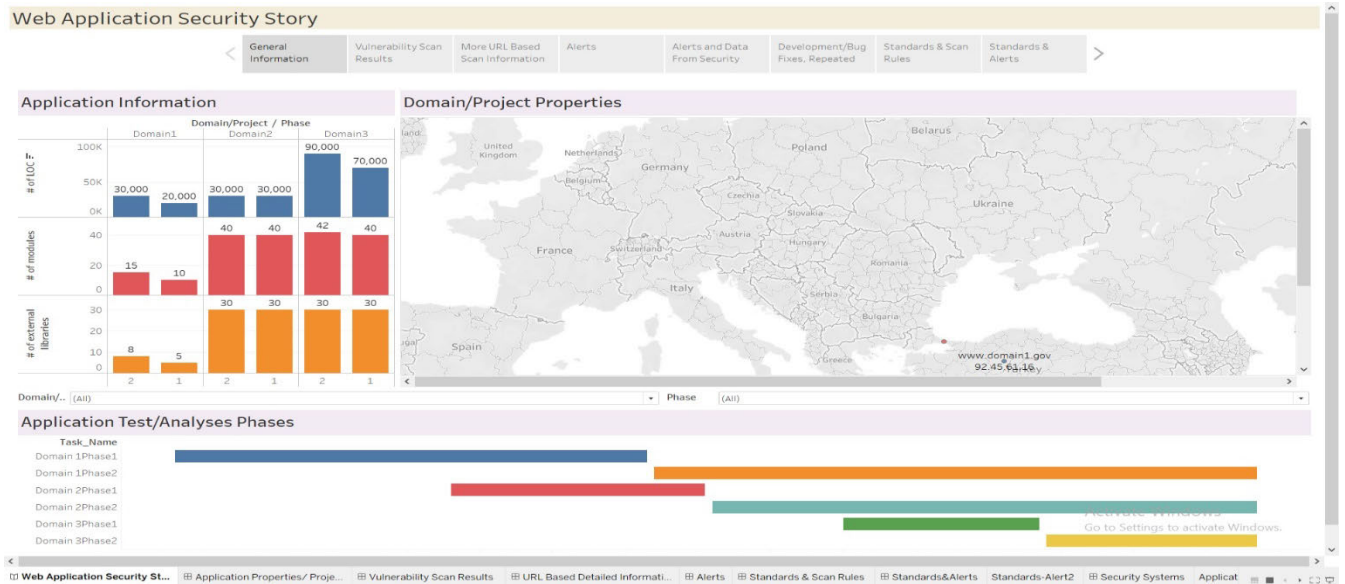


FIGURE 4. General information dashboard.

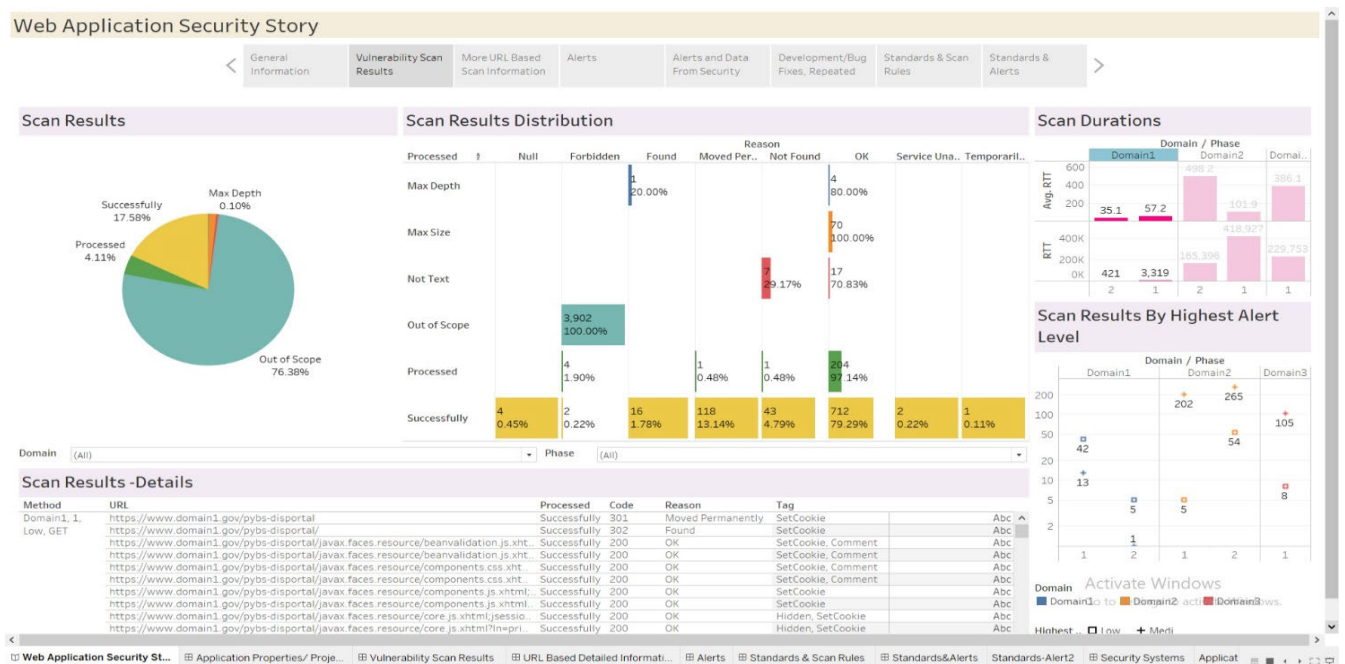


FIGURE 5. Vulnerability scan results dashboard.

very long time is given and a thorough investigation of the raw data is expected, the participants would not be willing to do that, and in the end, the majority would have opted out or left answers blank. Thus, the validation would yield incorrect results. If they are given a short time, this would still not be a fair comparison, because without having the HWAS-V tool, it is not possible to answer quiz questions which are related to multiple projects and phases in a short time, which would again result in blank responses. This would also

require involvement of the authors to the process to explain the relations and order of raw data to the users.

The responses, which are given to each scan effort, the distribution of successfully scanned and unscanned pages, the scan durations for each scan, detailed scan results, scan results by domains, and phases are shown in these views.

Fig. 4 to Fig. 11 illustrate different parts of the proposed dashboard-based tool prototype. Fig. 4 shows the first dashboard design which provides a top-level view of

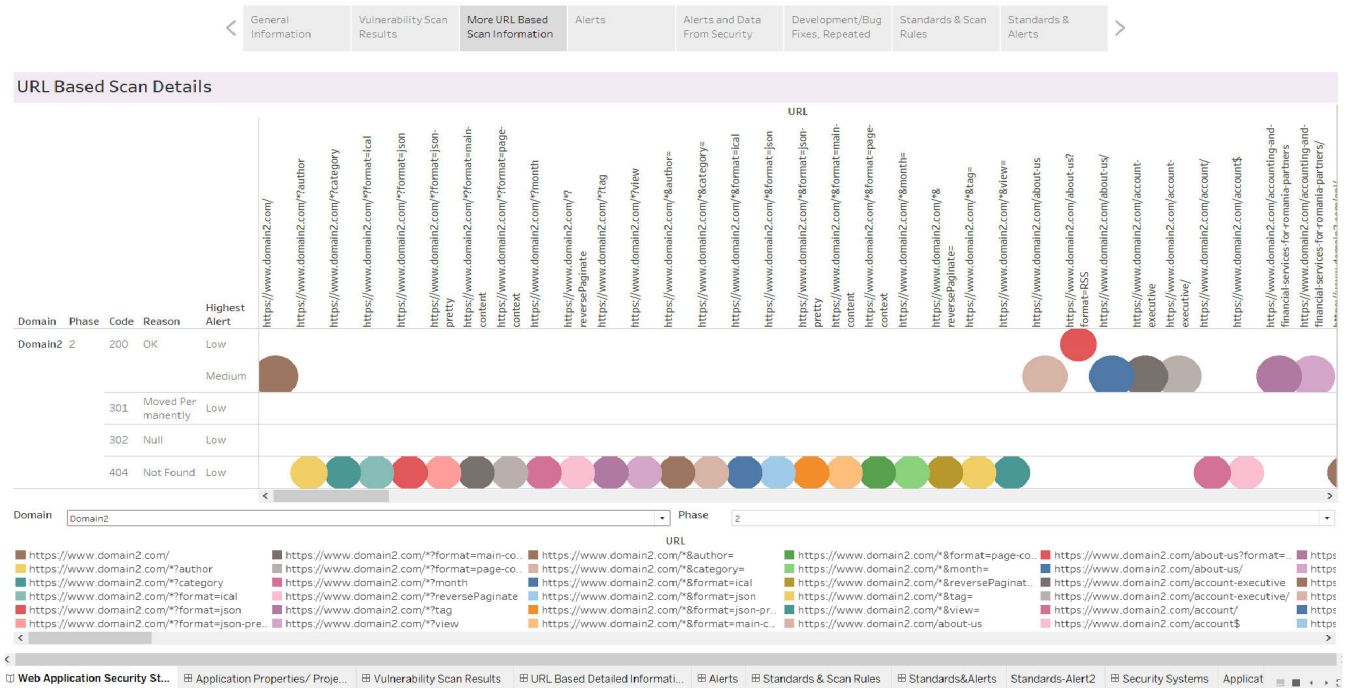


FIGURE 6. URL based scan details dashboard.

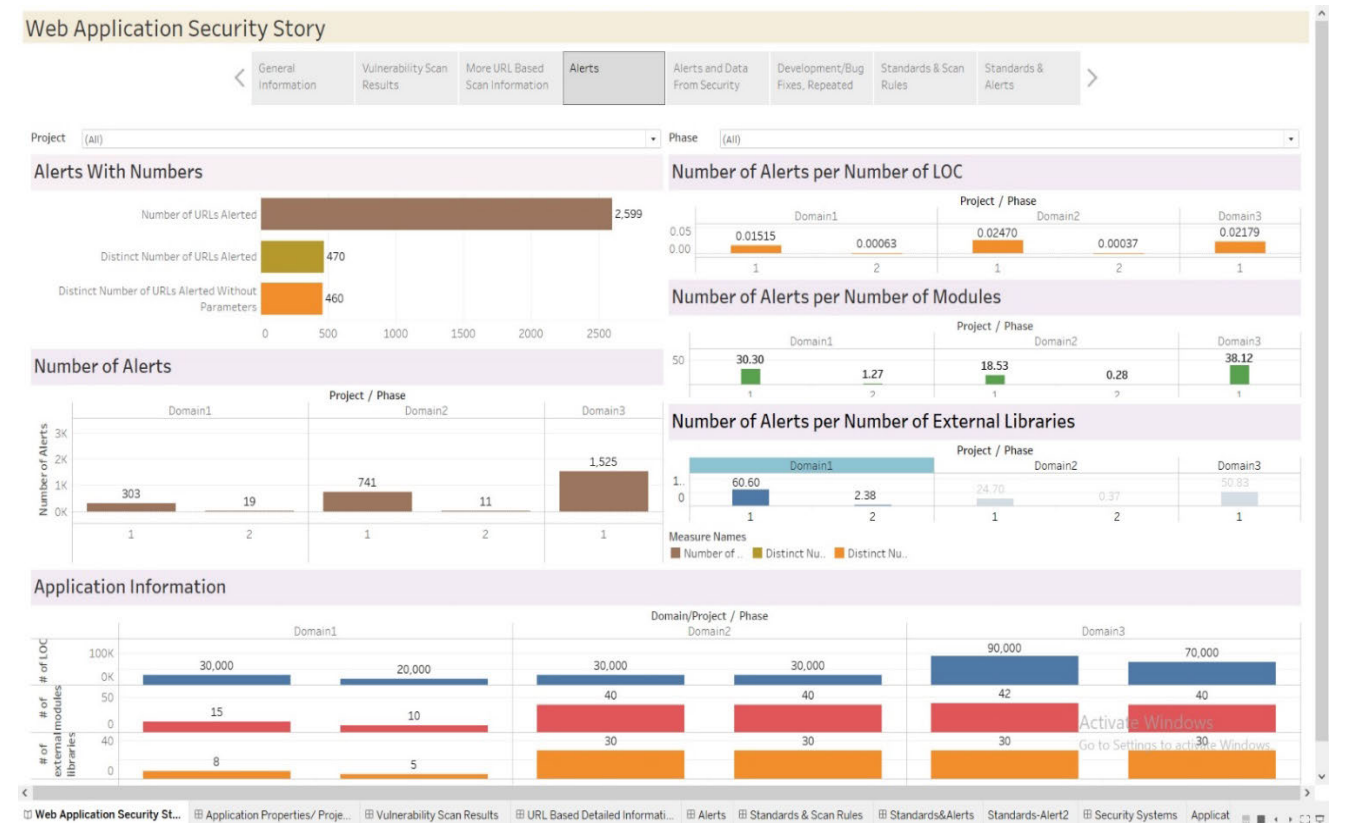


FIGURE 7. Alerts dashboard.

these three web applications' security statuses. The locations where the web pages are installed are shown on a map. This location information is useful for security analysts who monitor security statuses distributed in large regions.

Showing the location information would point out hosting place-based problems for multiple web projects distributed on multiple hosts. In this view, besides location, IP, and base URL, information related to web application size,

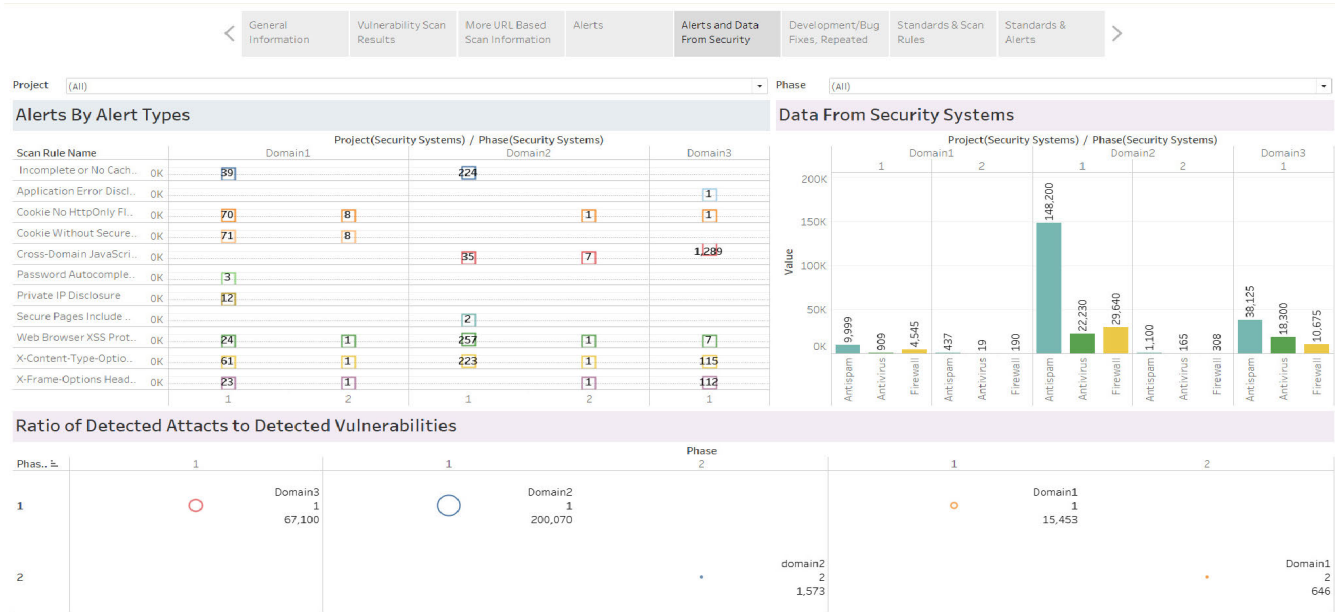


FIGURE 8. Alerts and data from security protection systems.

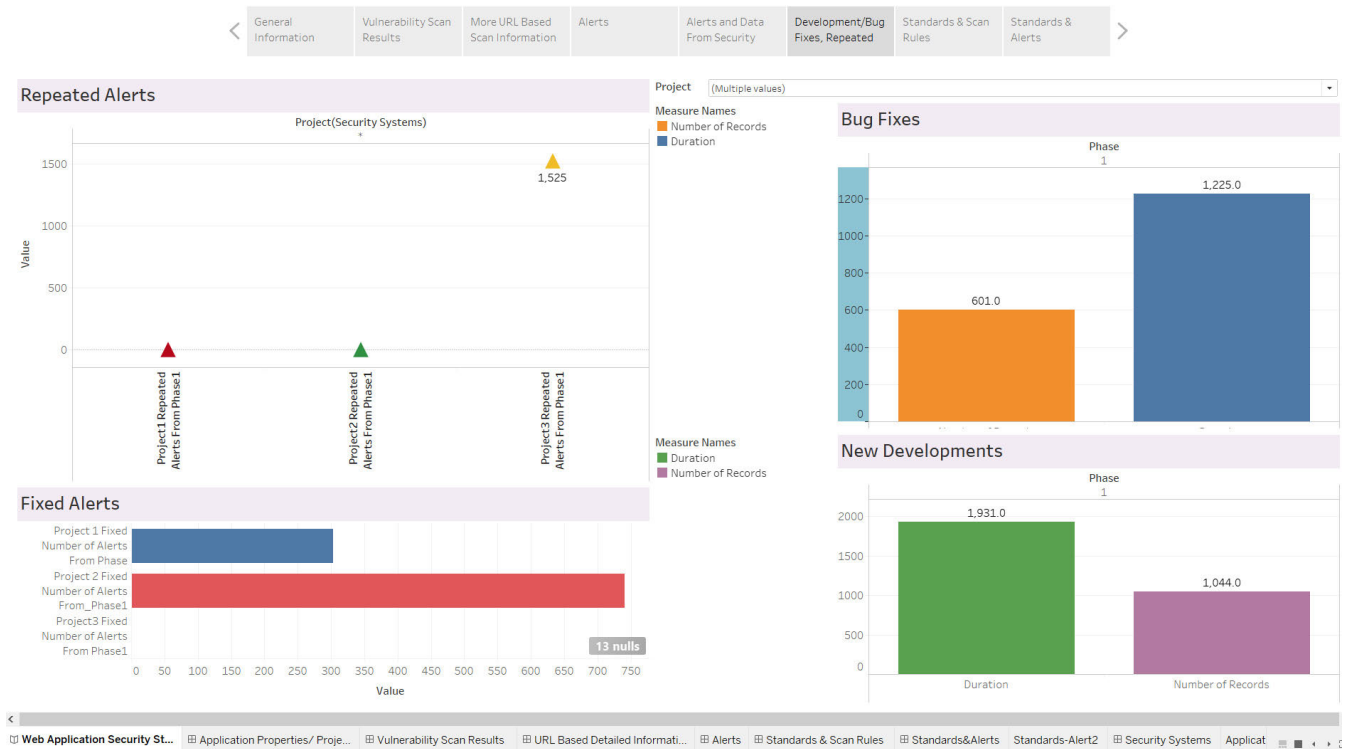


FIGURE 9. New developments, bug fixes, repeated alerts, fixed alerts.

and the earlier testing efforts are shown in a Gantt chart. Whereas in Fig. 5, the basic information related to the scan results is shown, Fig. 6 provides URL based detailed scan information. The responses, which are given to each scan effort, the distribution of successfully scanned and unscanned pages, the scan durations for each scan, detailed scan results,

scan results by domains, and phases are shown in these views.

A few of the charts are repeated among multiple dashboards due to relevancy, for example, application size related information is also included in Fig. 7, alerts dashboard. In this view, it is possible to see the number of alerts for each project,

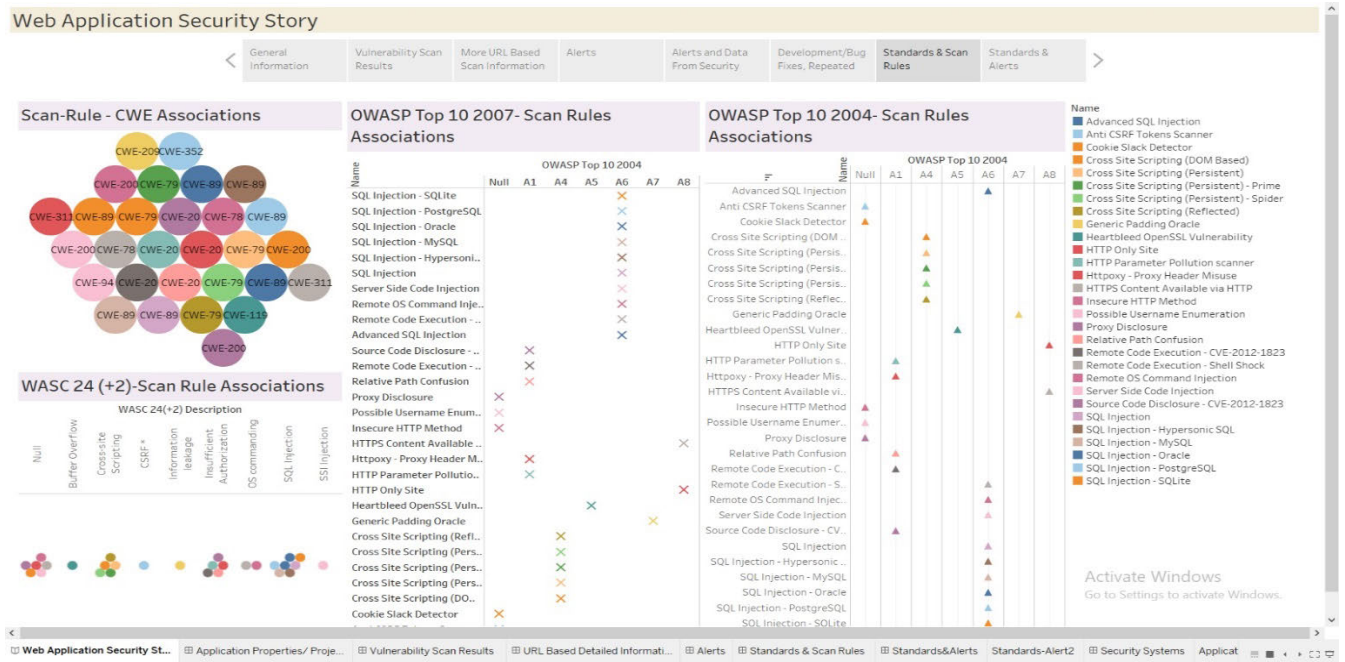


FIGURE 10. Standards and Scan Rules.

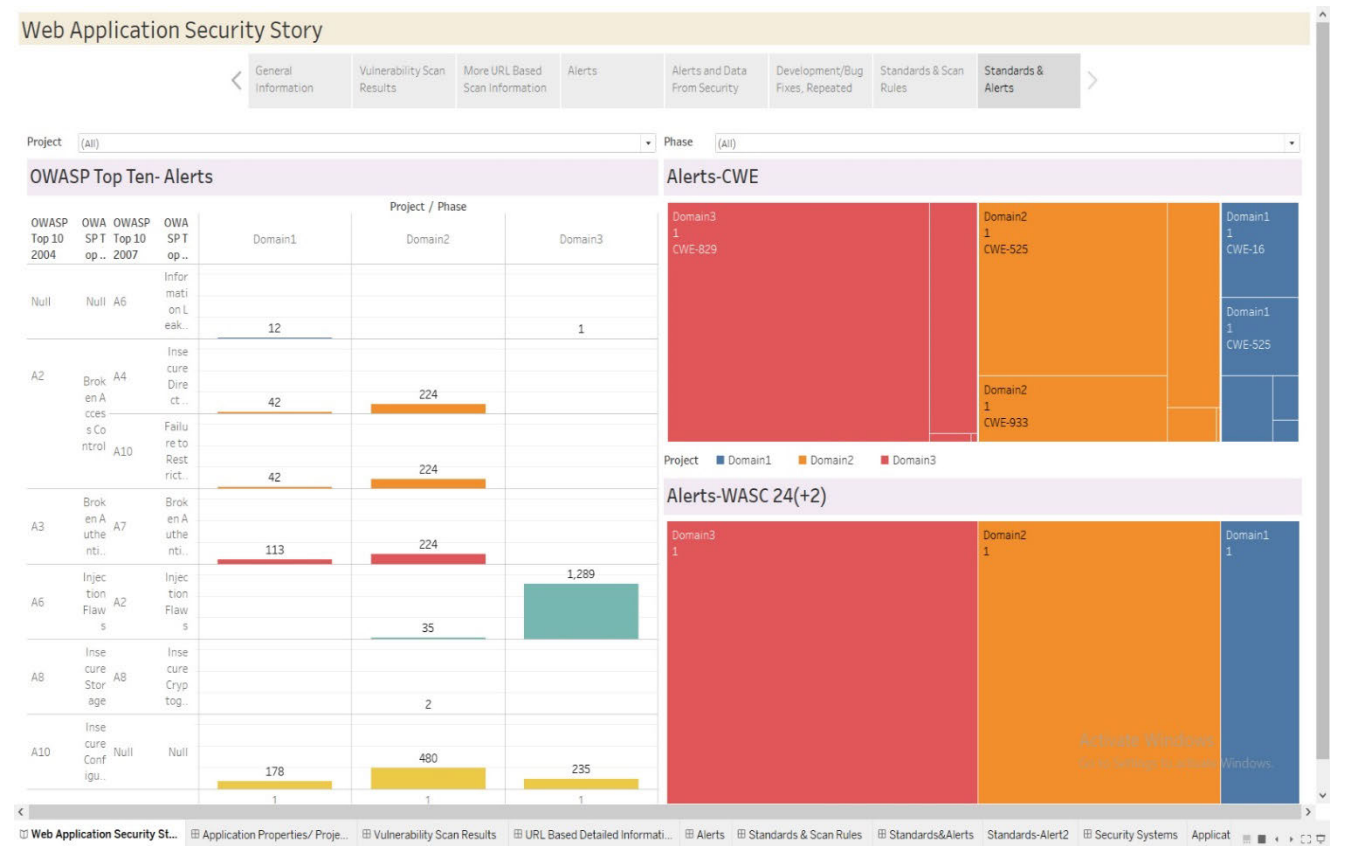


FIGURE 11. Standards and Alerts.

the distinct number of alerted URL's. Numerical information based on the number of modules, the number of external libraries, and the number of lines of code are available in this dashboard view.

In Fig. 8, alerts information is joined with the data coming from other existing security systems. In the environment where the web application is installed, there would be other protection systems, such as firewall, anti-spam, and anti-virus

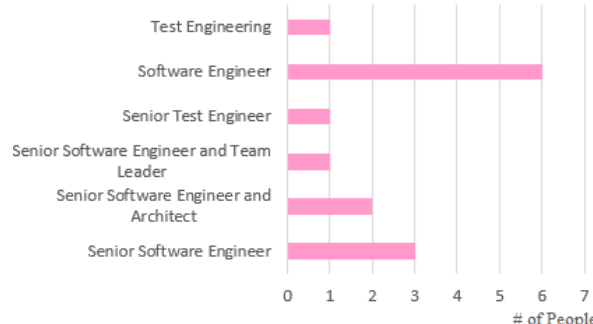
systems. In this view, the percentile of detected alerts by the vulnerability scanner tool and other security systems is also provided to increase understanding of the effects of vulnerability scans and security protection systems. The alerts classified by scan rules are also included in this view. The reason for presenting this information together is due to the fact that, some of the scan rules may be associated with protection systems due to their direct relation to the threats. For example, some threats are naturally better covered by firewalls, whereas some others by anti-virus systems.

When the sets of alerts from one phase are compared with sets of alerts from other phases through set operators, it is possible to find out sets of repeated alerts from the previous phase and fixed alerts from earlier phases for each project/domain. This will provide meaningful information related to the overall security status and efforts given for the web application concerning security. To empower the dashboard, the number of security-related new developments and bug fixes were also included in this view as shown in Fig. 9. In the data structure, all new developments and bug fixes are associated with a URL page. When the new development or bug fix is not directly related to a URL page, then the base URL can be used to obey the provided data structure. Typically, as mentioned before, web application vulnerability scanners' working mechanisms obey the black-box testing principals. Providing the secondary information selected for this study as part of the data structure would require internal knowledge for the application and project development process. If the mentioned secondary information, which would exist in a white-box testing environment, such as the information related to the application size, modules and project development process, then corresponding metrics will be available for the users. If such data is not available, HWAS-V will work with visualizing fewer metrics (the metrics related to non-existent data will always be blank), but overall data structure and screens will not be affected..

Fig. 10 shows the associations of scan rules to OWASP top ten 2004, OWASP top ten 2007, WASC 24(+2), and the CWE standards. In order to associate the scan rules to the case study data, web resources were used. These associations were stored as part of the data structure. The associations prepared for the case study do not cover all alert types and scan rules. Fig. 11 shows the distribution of the alerts to the OWASP, CWE, and WASC standards. Compatibility with the standards is valuable for most of the projects due to regulations or other obligations. Knowing the alerts related to the well-known vulnerabilities and working on them using recommended best practices would eventually end up with better security for web applications.

Using the case study data and outputs, a validation study was conducted. The participants of this validation study were recruited using authors' social and professional contacts. The characteristics of the participants are shown in Table 9. The aim of the study was described briefly to each participant while sharing a paper copy of the quiz and the survey to enable them to quickly determine if they are able or willing

TABLE 9. Expert evaluation participant characteristics.

Characteristic	Value
Number of participants	14
Profession	Software development, design and test engineers having a solid understanding and knowledge of web application security.
	
Average age	30.786
Min age	25
Max Age	38
Std Age	4.492
Average years of experience	7.214
Min years of experience	1
Max years of experience	15
Std years of experience	4.492
Gender	21.43% Female, 78.57% Male

to assist the study. Later, electronic versions of the quiz and survey were shared to collect actual data. As described in the process description part, the validation study includes a quiz that should be answered using the HWAS-V. After this quiz, a short survey was conducted to query the practicality, efficiency, decision informing, and difference detection attributes of the proposed system. The quiz questions and the number of correct and incorrect answers are presented in Table 10. Numerical evaluation results achieved related to HWAS-V features are provided in Table 11.

One-tailed t-test was used to test the significance of the survey outputs. The hypothesis was "The applicants found that the proposed tool was more than "Helpful" (numeric value =3 in Likert scale)" for the chosen measurements, namely practicality, efficiency, decision informing, and difference detection. The p results were 0.16, 0.78, 0.01, and 0.01 for practicality, efficiency, decision informing, and difference detection properties, respectively. Considering a critical value of 0.05, the evaluation results show that there is enough evidence to infer that "Decision Informing" and "Difference Detection" properties of the proposed design is significantly greater than "Helpful" according to the users. However, the t-test fails to reject the null hypothesis that the mean is 3 for "Practicality", and "Efficiency". Their means were over the "Slightly Helpful" region. According to the authors, the par-

TABLE 10. Evaluation questions and results.

Question	T	F
1-In which city does the web application corresponding to "domain 3" locates? a) Ankara, b) Istanbul, c) New York, d) Other	14	0
2-For all three projects two vulnerability measurements were done. Select the most vulnerable software project based on the number of vulnerabilities per project size measured as LOC (line of code). a) Domain1, b) Domain2, c) Domain3, d) All are equal	2	12
3-For all three projects two vulnerability measurements were done. Select the project for which no new development or bug fix was done between two analyses phases? a) Domain1, b) Domain2, c) Domain3, d) All are equal	11	3
4-What is the number of pages which have highest associated alert "Low" for "domain1" web application project? a) 105, b) 42, c) 265, d) 13	7	7
5-Vulnerability scanners can not process all the pages for web applications due to several reasons. One of these reasons is reaching the "Max Depth". Looking at the overall results for all three projects, what is the percentage of pages which are unprocessed due to reaching max depth. a) 10%, b) 0.1%, c) 20%, d) Other	12	2
6-Which project has the highest round-trip time taken for a vulnerability scan session? a) Domain1, b) Domain2, c) Domain3, d) All are equal	14	0
7-What is the "metric name" shown in the tool tip box for the previous dashboard a) OWASPTopTen2007Vulnerabilities, b) Number of Vulnerabilities, c) URLProcessedSet, d) URLsWithAlert	12	2
8-What is the number of alerts per modules per "Phase 2" of "Domain 2"? a) 0.28, b) 38.12, c) 100, d) 60.60	14	0
9-For which project, the project size did not change between two independent vulnerability scans? a) Domain 1, b) Domain 2, c) Domain 3, d) None	11	3
10-For "Domain 1" in "Phase 1", what is the number of vulnerabilities of the type "Web Browser XSS Protection Not Enabled"? a) 24, b) 1, c) 7, d) 35	14	0
11-What is the number of repeated alerts for "Domain 3" in "Phase 2"? a) 0, b) 1525, c) 303, d) 741	8	6
12-What is the number of fixed alerts for "Domain 2" in "Phase 2" from "Phase 1"? a) 0, b) 1525, c) 303, d) 741	11	3
13- Based on the scanner rules used in this tool, how many vulnerabilities in the CWE database were covered? a) 50, b) 27, c) 4, d) 0	9	5
14-Is "A10 - Failure to Restrict URL Access" of OWASP top ten 2007 vulnerabilities list is covered in the existing scanner rules? a) Yes, b) No, c) I don't know, d) N/A	7	7
15- For "Domain 2", what is the number of vulnerabilities of the type "A6 Injection Flaws"? a) 35, b) 2, c) 224, d) 71	14	0

TABLE 11. Summary of feature T-test results.

	<i>Practicality</i>	<i>Efficiency</i>	<i>Decision Informing</i>	<i>Difference Detection</i>
Mean	2.71	2.92	3.64	3.71
Observations	14	14	14	14
Hypothesized Mean	3.0	3.0	3.0	3.0
Df	13	13	13	13
sd	0.73	0.92	0.84	0.91
SE	0.19	0.24	0.22	0.24
t-stat	1.47	0.29	2.86	2.92
P	0.16	0.78	0.01	0.01

participants were quite successful in answering a relatively complicated set of questions with numerous comparisons and a high level of decision information in a reasonable time.

VI. USAGE SCENARIOS FOR DIFFERENT ROLES

In this section, the demonstration of the usage of the tools for six different roles is made through flowcharts in Fig. 12 and Fig. 13. The selected roles include executives, developers, project managers, network/system administrators, security

auditors, and security analysts. All of these roles have their own level of interest in the vulnerability data and their own tasks and duties. The designed system allows conducting these tasks in the most efficient manner. The majority of these roles can achieve their specific purposes by using at most two tabs from the visualization system. The project manager and executive have to deal with three different dashboards. This shows that although the number of graphics and metrics is very high, the overall design results in the

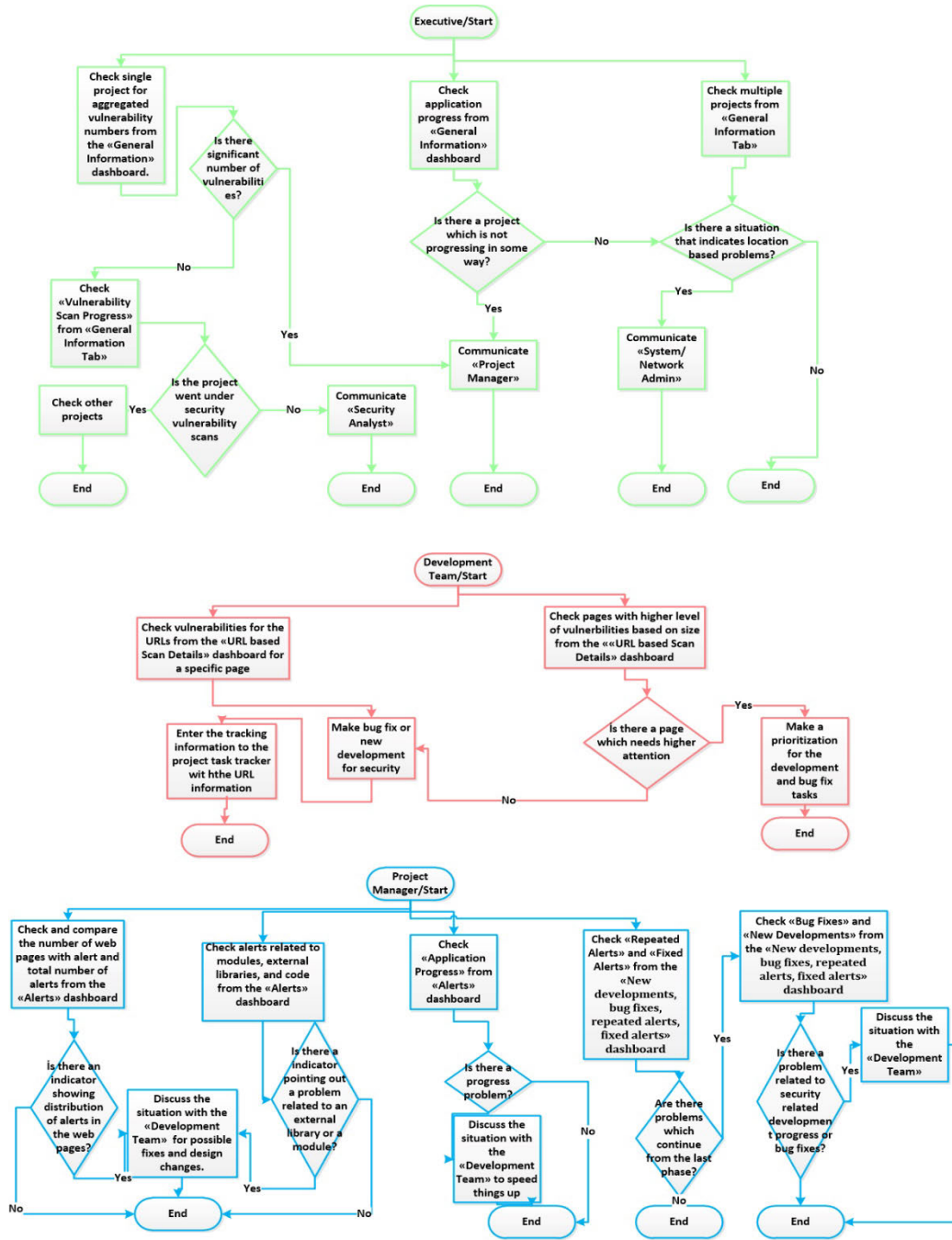


FIGURE 12. Workflow for different user prototypes- Part1.

efficient processing of vulnerability data. In fact, although the provided vulnerability data is very detailed and more connected with other project information, web application vulnerability related tasks became very straightforward for all roles due to grouping, separation, and ordering of the metrics and corresponding visualizations.

- Executive: “General Information” dashboard
- Developer: “URL based Scan Details” dashboard
- Project Manager: “Alerts” dashboard, “New developments, bug fixes, repeated alerts, fixed alerts” dashboard

- Network/System Admin: “Alert by Alert Types” dashboard, “Data from Security Systems” dashboard
- Security Auditor: “Standards and Scan Rules” dashboard, “Standards and Alerts” dashboard
- Security Analyst: “Vulnerability Scan Results”

VII. DISCUSSION

This study examined common outputs of web application security vulnerability scanner tools and provided a data structure that is further used during the definition of a set of metrics and measures. New metrics were defined

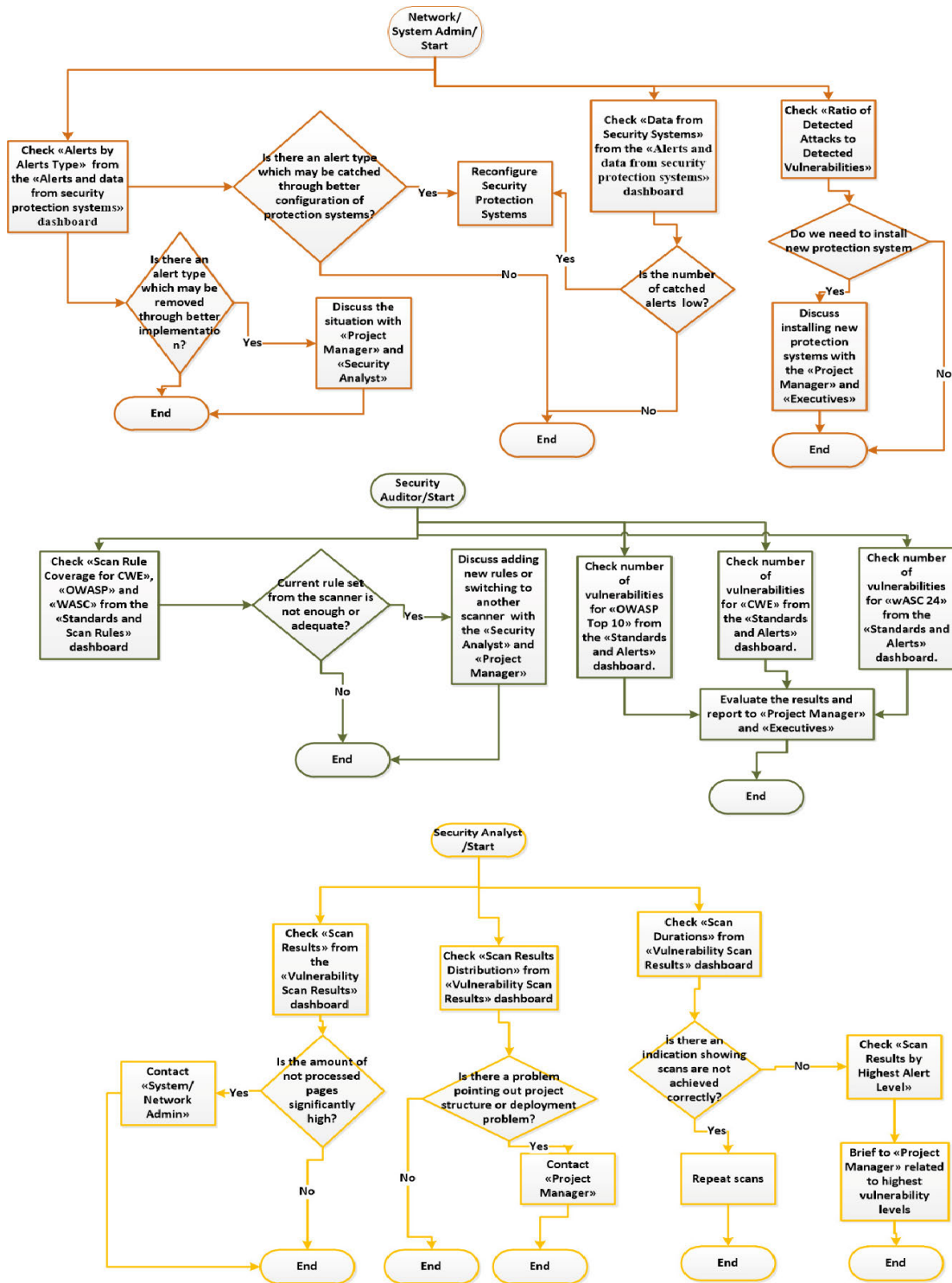


FIGURE 13. Workflow for different user prototypes—Part 2.

by combining the initial set of standard web application security metrics with possible effects of secondary data sources which may be originated from the development efforts, web application properties, or the dynamics of the system environment. A case study was presented

showing the visualizations based on data generated using OWASP Zed Attack Proxy (ZAP) tool together with some user-generated sample data as an improvement to the existing web application security vulnerability reporting systems.

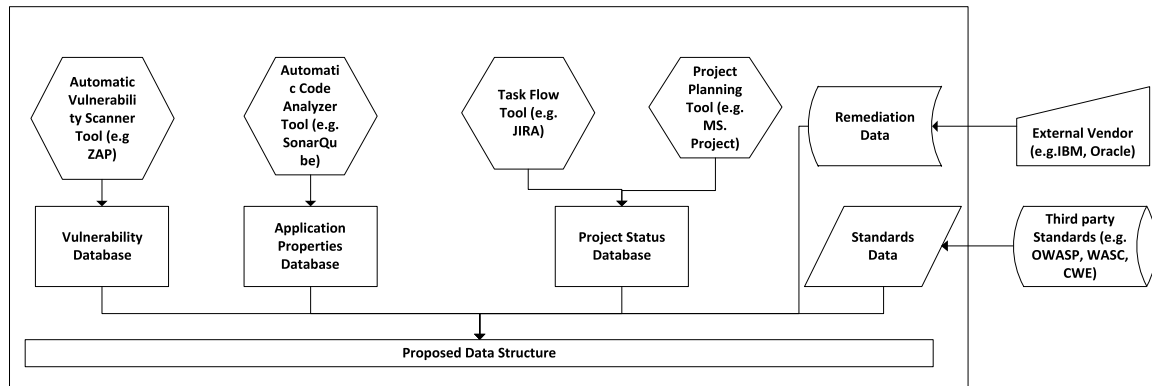


FIGURE 14. A conceptual model to form the proposed data structure.

Few studies focus on the visualization of web application vulnerability scan results in the security visualization domain [14]. The only existing study which targets web application security black-box test results enables visualization of statistical measures. The measures/metrics proposed in this study would enable a broader perspective of the security status for various stakeholders. More studies are required in this area to empower an extensive comparative analysis for this field.

Contributions of this study include a new dashboard tool for visualizing vulnerability scan results based on a unique data structure formed by combining multiple data sources. Using the phasing structure allows combining these multiple data sources. The design was developed through the use of Tableau software. Tableau dashboard designs can be viewed through Tableau Public, Tableau Desktop, and Tableau Reader applications. They may also be integrated with any web application which allows frame-based HTML pages. Using Tableau Public will not be appropriate for viewing the web vulnerability scanner results for security reasons. However, Tableau Reader and Tableau Desktop may be more adequate to access HWAS-V. The secondary contribution of this work is the list of metrics/measures that the tool presents. The paper also presents a case study and the validation efforts.

Some of the proposed metrics/measures were left out during the prototype design. Collecting information related to the average remediation latencies for each alert type was left as future work. Similarly, the classification of the alerts based on their effects on sensitive information, their impacts on business, and their relation to the existing vulnerable components was also left out.

The main limitation of this study is the use of OWASP Zed Attack Proxy tool for the case study. The proposed system was not tested with other vulnerability scanner outputs. The list of available attributes may change slightly using other vulnerability scanners. The proposed metrics provide information related to the various aspects of web application security. It enables monitoring and comparing independent analyses for multiple projects. It is not limited to the raw outputs of the vulnerability scanner. On the contrary, it serves a quite large number of metrics and measures. However, there

are some concepts which the proposed metrics are not directly related.

The web application security-related factors which are not measured with the proposed metrics/measures are:

- Tool efficiency
- Number of false positives
- Number of false negatives
- Security economics
- Cost to fix
- The success of security education/certification
- Defect injection ratio
- The success of code analysis
- Defect detection ratio during code analysis
- The success of the test
- Defect detection ratio by testers

In order to measure tool efficiency, ZAP results should be compared to manual inspection or similar results or should be compared with the results of other tools. However, such a comparison, thus, measuring the tool efficiency, was not in the scope of this work. In order to measure the security economics, various other data types, such as precaution costs, personnel costs and education costs should be associated with the ZAP data. This association, and, thus, security economics of web applications was not in the scope of this work. The success of security education was also completely outside the scope. Presenting the success of the users of the tool would require a comparison among multiple users, or a test project with known defects, which allows measuring the success of the users.

During the study, the Tableau software was evaluated to some extent for its suitability to develop a dashboard based on data coming from multiple sources and showing a large number of metrics in association with each other. The results showed that the software was a proper choice to design and develop complex security dashboards with feasible effort and time.

As mentioned previously, due to having a dashboard design, and including quite a large number of metrics, HWAS-V can be considered as a Security Information and

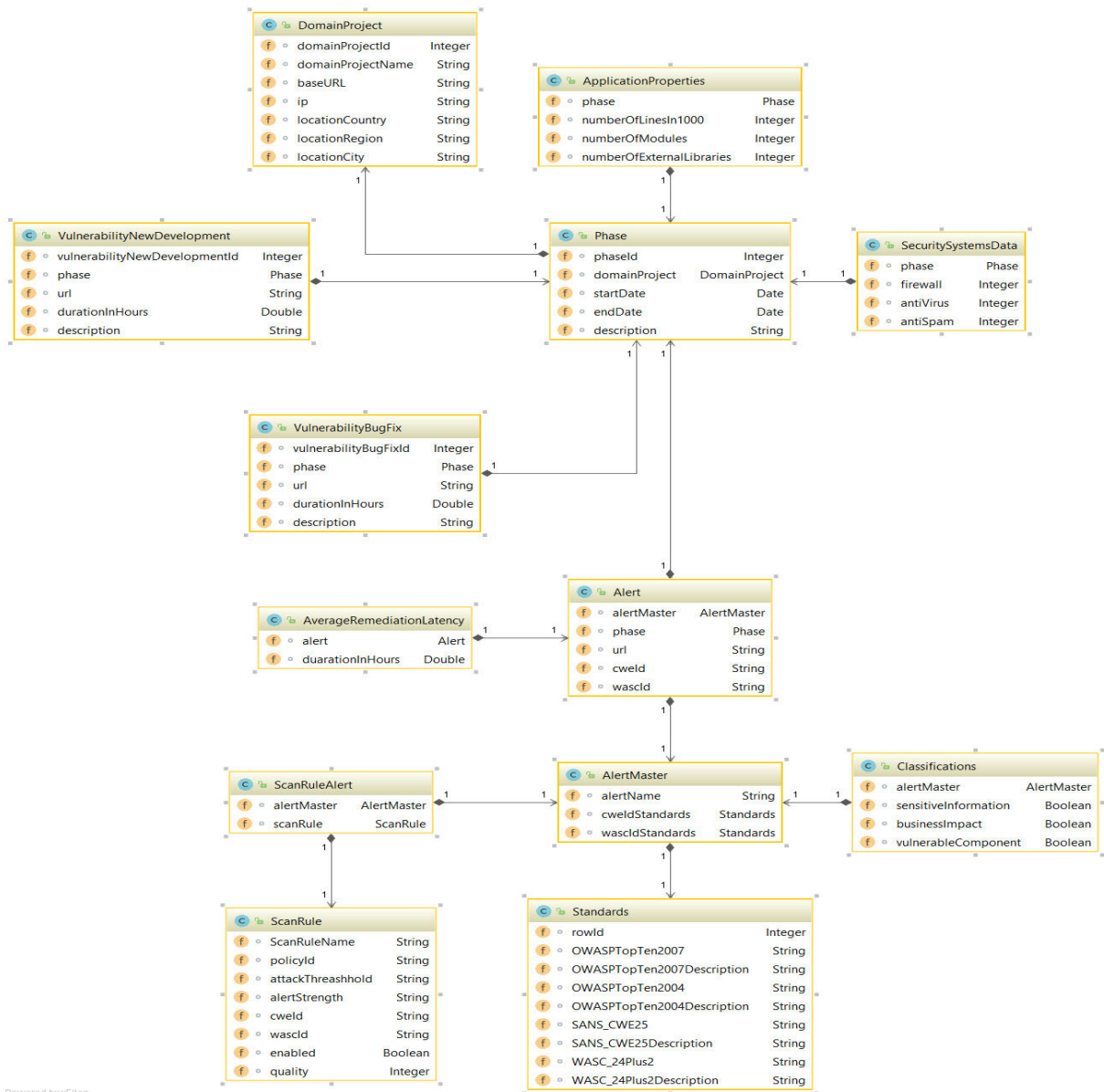


FIGURE 15. Data structure and attributes of the proposed model.

Event Management (SIEM) tool and can be compared to them. Gartner [35] divided the SIEM products into four quadrants; Leaders, Challengers, Visionaries, and Niche Players. Based on this categorization HWAS-V can be located in the Niche players region due to its niche focus area, vulnerability scan results of black-box vulnerability scanners.

A critical factor that differentiates the HWAS-V from the SIEM tools is the decision of joining project life cycle related information with the vulnerability scan results. This differentiation also exists due to integrating the vulnerability scan results with multiple security standards, not just one, and having a built-in structure allowing comparison of vulnerability scan results in multiple phases collected intermittently due to set operations. HWAS-V has a project management

perspective. It aims to provide a way to monitor the security-related progress, such as new developments and bug fixes associated with previous alerts that do not commonly exist in the SIEM systems.

Investigation results indicate that SIEM tools and the proposed web application vulnerability visualization tool are prominently different, since both their intended purposes and features do not overlap. The intended purpose of SIEM tools is to provide ways to collect data, to analyze data in real-time, generate compliance and regulatory reports, to correlate data and to find out indicators of events, and to present these findings. However, the intended purpose of HWAS-V is to provide an efficient way to examine present, past, and recent vulnerability scan results that are coming from black-box

tests for one or more projects for the decision informing and difference detection purposes.

VIII. CONCLUDING REMARKS

This paper presented a new visualization study focusing on web application vulnerability scanner results. The visualization supports a large set of measures/metrics. It integrates the vulnerability scanner data with some secondary data sources. In this way, it provides both a technical view and a managerial view. A visualization tool called HWAS-V was developed (available under this name on GitHub) and demonstrated to the participants, who then evaluated the tool.

The results indicate that the proposed design can help analysts and managers due to its “decision informing” and “difference detection” capabilities. Its level of “Efficiency” and “Practicality” were, on the other hand, questionable. These were in the “Slightly Helpful” range based on the validation results. The large number of metrics and the large number of charts distributed to eight interconnected dashboards might affect the level of “Efficiency” and “Practicality” of the proposed design. A lighter version of the proposed design with a smaller number of charts and metrics might have different results.

In the authors’ best knowledge, dashboard security visualization systems with a set of metrics/measures involving a specific user interaction with precise identification of each metric in the displays via tooltips is unique to this study, which has not been used in security visualization systems before. Precise identification of the metrics will increase the usability of the proposed system.

The system will provide a broad perspective of the security status of one or more projects. It also allows presenting results from subsequent analyses made by automatic web security analysis tools and comparison among them. Automatic comparison of subsequent scans will also enable understanding if there is a barrier that prevents proper scanning in a specific vulnerability analysis session. The proposed system was demonstrated using data generated by the ZAP tool. Incorporating other automated web vulnerability scanner results seems like a logical direction for future research.

APPENDIX

A. A CONCEPTUAL MODEL TO FORM THE PROPOSED DATA STRUCTURE

In this part a conceptual model presenting possible integrations of different systems to form the proposed data structure is presented. The user need not adhere to these sample software products. They may use manual ways or replace some software with other alternatives.

B. DATA STRUCTURE USED AND PROPOSED IN THIS STUDY

The subparts of these groups and their relationships are provided in Figure 15. DomainProject records can be manually inserted into the database once for each web application project. Phase table stores the records that define each vul-

nerability scan phase and can be manually entered when a set of scan results are uploaded to the system. ApplicationProperties records can also be entered manually as new phases, i.e., new vulnerability scans, occur. The values in this table may be static or may change if the size and/or components of the web project changes. VulnerabilityNewDevelopment and VulnerabilityBugFix tables contain project development’s related data. Similar to ApplicationProperties records, these records can be created either manually or automatically, if integration is possible. Some task management tools, such as JIRA may allow querying the number of task records belonging to each category between the phase start and end dates. AlertMaster, ScanRule and ScanRuleAlert relation tables hold the metadata. This part contains the only tool dependent data part of the proposed visualization tool. Each vulnerability scanner has its own set of scan rule names and corresponding alert names. However, according to the general state of the art, scan rules and alerts always exist. If the user wants to use other scanner tools, then corresponding metadata should be created. Once this metadata is created, corresponding records that hold their relationships to the Classifications and the Standards should also be entered to the Standards and Classifications tables. The structure of the main table, Alerts table, is very generic indeed, holding the id, associated URL, and phase_ID of the alert found. Thus, it should not differ much from one tool to another. The execution of the proposed tool mainly depends on having the metadata definitions, alerts data, and the temporal data which defines the application/project properties independent of the scanner tool. The proposed structure can be used for scan results from different scanners, as long as the metadata prepared for each scanner has consistent names for the same alerts/scan rules and different names for different alert/scan rules without causing any collision and conflict.

REFERENCES

- [1] K. A. Demir, “A survey on challenges of software project management,” in *Software Engineering Research and Practice*. Las Vegas, NV, USA: Stylus Publishing, 2009.
- [2] B. Molnar and A. Tarcsi, “Architecture and system design issues of contemporary web-based information systems,” in *Proc. 5th Int. Conf. Softw., Knowl. Inf., Ind. Manage. Appl. (SKIMA)*, Benevento, Italy, Sep. 2011, pp. 1–8.
- [3] Imperva. *The State of Web Application Vulnerabilities in 2019*. Accessed: Jan. 23, 2020. [Online]. Available: <https://www.imperva.com/blog/the-state-of-vulnerabilities-in-2019/>
- [4] Web Application Security Testing Fundamentals. (2021). *Software Testing Tips and Tricks*. Accessed: Jan. 15, 2021. [Online]. Available: <https://www.softwaretesttips.com/web-application-security-testing/>
- [5] Acunetix. *Acunetix*. Accessed: Jan. 8, 2019. [Online]. Available: <https://www.acunetix.com/>
- [6] Netsparker | *Web Application Security Scanner*. Netsparker. Accessed: Jan. 8, 2019. [Online]. Available: <https://www.netsparker.com/>
- [7] OWASP. *The OWASP Foundation*. Accessed: Nov. 5, 2018. [Online]. Available: https://www.owasp.org/index.php/Main_Page
- [8] Portswigger. *Portswigger Web Security-BurpSuite*. Accessed: Jan. 8, 2019. [Online]. Available: <https://portswigger.net/>
- [9] O. Romania. *OWASP Zed Attack Proxy*. Accessed: May 20, 2018. [Online]. Available: https://www.owasp.org/images/9/96/OWASP_2014_OWASP_ROMANIA.pdf

- [10] M. Bingham, A. Skillen, and A. Somayaji, "Even hackers deserve usability: An expert evaluation of penetration testing tools," in *Proc. 9th Annu. Symp. Inf. Assurance*, Albany, NY, USA, 2014, pp. 1–95.
- [11] S. Chiasson, R. Biddle, and A. Somayaji, "Even experts deserve usable security: Design guidelines for security management systems," in *Proc. Symp. Usable Secur. Privacy (SOUPS) Workshop Usable e Secur. Privacy (SOUPS) Workshop Usable IT Secur. Manage.*, 2007, pp. 1–4.
- [12] J. R. Nurse, S. Creese, M. Goldsmith, and K. Lamberts, "Guidelines for usable cybersecurity: Past and present," in *Proc. 3rd Int. Workshop Cyberspace Saf. Secur. (CSS)*, Milan, Italy, Sep. 2011.
- [13] F. Özdemir Sönmez, "Security visualization infrastructures, techniques, methodologies for improved enterprise security," Ph.D. dissertation, Graduate School Inform., METU, Ankara, Turkey, 2019.
- [14] F. Ö. Sönmez and B. Günel, "Security visualization extended review issues, classifications, validation methods, trends, extensions," in *Security and Privacy Management, Techniques, and Protocols*. Hershey, PA, USA: IGI Global, 2018, pp. 152–197.
- [15] T. T. Dang and T. K. Dang, "An extensible framework for Web application vulnerabilities visualization and analysis," in *Future Data and Security Engineering*. Ho Chi Minh City, Vietnam: Springer, 2014, pp. 86–96.
- [16] H. Assal, S. Chiasson, and R. Biddle, "Cesar: Visual representation of source code vulnerabilities," in *Proc. IEEE Symp. Vis. Cyber Secur. (VizSec)*, Baltimore, MD, USA, Oct. 2016, pp. 1–8.
- [17] J. R. Goodall, H. Radwan, and L. Halseth, "Visual analysis of code security," in *Proc. 7th Int. Symp. Visualizat. for Cyber Secur. (VizSec)*, New York, NY, USA, 2010, pp. 46–51.
- [18] L. Harrison, R. Spahn, M. Iannacone, E. Downing, and J. R. Goodall, "NV: Nessus vulnerability visualization for the Web," in *Proc. 9th Int. Symp. Vis. Cyber Secur.*, 2012, pp. 25–32.
- [19] M. Alsaleh, A. Alarifi, A. Alqahtani, and A. Al-Salman, "Visualizing Web server attacks: Patterns in PHPIDS logs," *Secur. Commun. Netw.*, vol. 8, no. 11, pp. 1991–2003, 2015.
- [20] T. T. Dang and T. K. Dang, "Visualizing Web attack scenarios in space and time coordinate systems," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XVI*. Ho Chi Minh City, Vietnam, 2014, pp. 1–14.
- [21] D. Gugelmann, F. Gasser, B. Ager, and V. Lenders, "Hviz: HTTP(S) traffic aggregation and visualization for network forensics," in *Proc. 2nd Annu. DFRWS Eur.*, 2015, pp. S1–S11.
- [22] Z. Kan, C. Hu, Z. Wang, G. Wang, and X. Huang, "NetVis: A network security management visualization tool based on treemap," in *Proc. 2nd Int. Conf. Adv. Comput. Control*, Mar. 2010, pp. 18–21.
- [23] B. Johnson, Y. Song, E. Murphy-Hill, and R. Bowdidge, "Why don't software developers use static analysis tools to find bugs?" in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, San Francisco, CA, USA, May 2013, pp. 672–681.
- [24] Zap proxy. ZAP OWASP Zed Attack Proxy. Accessed: Jan. 8, 2019. [Online]. Available: <https://www.zaproxy.org/>
- [25] A. Meneely, B. Smith, and L. Williams, "Validating software metrics: A spectrum of philosophies," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, p. 24, 2013.
- [26] J. Nielsen, *10 Usability Heuristics for User Interface Design*, Nielsen Norman Group, Fermont, ON, Canada, 1995.
- [27] D. G. Murray, *Tableau Your Data!: Fast and Easy Visual Analysis With Tableau Software*. Indianapolis, IN, USA: Wiley, 2013.
- [28] D. T. Tri and T. K. Dang, "Security visualization for peer-to-peer resource," *Int. J. Comput. Sci. Eng.*, vol. 1, no. 2, pp. 47–55, 2009.
- [29] ManageEngine. 19 9 2018. Accessed: Sep. 19, 2018. [Online]. Available: <https://www.manageengine.com/>
- [30] Splunk. (Jul. 15, 2013). *JQuery Sparklines*. Accessed: Aug. 7, 2018. [Online]. Available: <https://omnipotent.net/jquery.sparkline>
- [31] Rapid7. *InsightIDR*. Accessed: Sep. 19, 2018. [Online]. Available: <https://www.rapid7.com/products/InsightIDR>
- [32] Solarwinds. *Solve Your Toughest IT Management Problem, Today*. Accessed: Sep. 19, 2018. [Online]. Available: <https://www.solarwinds.com/>
- [33] MicroFocus. *ArcSight Enterprise Security Manager*. Accessed: Sep. 19, 2018. [Online]. Available: <https://software.microfocus.com/en-us/products/siem-security-information-event-management/overview>
- [34] AlienVault. *AlienVault Unified Security Management*. Accessed: Sep. 19, 2018. [Online]. Available: <https://www.alienvault.com/products>
- [35] M. Nicolett and K. M. Kavanagh, "Magic quadrant for security information and event management," Gartner, Stamford, CT, USA, Tech. Rep. G00246886, 2013.
- [36] F. Ö. Sönmez and B. Günel, "Evaluation of security information and event management systems for custom security visualization generation," in *Proc. Int. Congr. Big Data, Deep Learn. Fighting Cyber Terrorism (IBIGDELFT)*, Dec. 2018, pp. 38–44.
- [37] IT Central Station. (2020). *Application Security Testing (AST)*. Accessed: Dec. 1, 2010. [Online]. Available: <https://www.itcentralstation.com/categories/application-security-testing-ast>
- [38] G. A. Campbell and P. P. Papapetrou, *Sonarqube in Action*, Manning Publications Co., Greenwich, CT, USA, 2013.
- [39] Parasoft. Accessed: May 20, 2018. [Online]. Available: <http://www.parasoft.com/>
- [40] D. Vitek, *Auditing Code for Security Vulnerabilities with CodeSonar*. Boston, MA, USA: IEEE, 2016.
- [41] N. Imtiaz, A. Rahman, E. Farhana, and L. Williams, "Challenges with responding to static analysis tool alerts," in *Proc. IEEE/ACM 16th Int. Conf. Mining Softw. Repositories (MSR)*, Montreal, QC, Canada, May 2019, pp. 245–249.



FERDA ÖZDEMİR SÖNMEZ (Member, IEEE) received the B.Sc. degree in electrical and electronics engineering, and the M.Sc. and the Ph.D. degrees in information systems from Middle East Technical University (METU), Ankara, Turkey, in 1997, 2012, and 2014, respectively. After receiving her B.Sc. degree, she worked in the private sector for 18 years as a Software Specialist, a Software Development Consultant, a Project Manager, and an IT Manager. She has been holding the PMP degree, since 2009. She worked on projects were mainly in the areas of e-government and telecommunications. She is currently a Postdoctoral Researcher with the Institute for Security Science and Technology, Imperial College, London, U.K. Her research interests include security visualization, security requirements engineering, and blockchain security.



BANU GÜNEL KILIÇ (Member, IEEE) received the B.Sc. degree (Hons.) in electrical and electronics Engineering from Middle East Technical University (METU), Ankara, Turkey, in 2000, the M.Sc. degree (Hons.) in communication systems and signal processing from the University of Bristol, U.K., in 2001, and the Ph.D. degree in computer science from the Queen's University of Belfast, U.K., in 2004. From 2004 to 2010, she worked as a Postdoctoral Researcher with the Center for Communication Systems and Signal Processing, University of Surrey, U.K. She is currently an Associate Professor with the Department of Information Systems, METU Graduate School of Informatics. Her research interests include signal processing, social network analysis, and security and surveillance.

...