A Common Pentest Output Schema for Business Intelligence System Ingestion

Shivangi Sharma, Justin M. Pelletier, Bill Stackpole
Rochester Institute of Technology
Rochester, NY 14623
Email: jxpics@rit.edu

***Abstract*—Data-driven Business Intelligence process improvements demand proactive digital vulnerability discovery and exploitation enumeration. These risk discovery and analyses practices are commonly known as penetration tests (pentests), and have emerged as requirement for most organizations under many compliance regimes such as the global Payment Card Industry Data Security Standard (PCI-DSS) and the federal Gramm-Leach Bliley Act (GLBA) in the United States. Though a growing body of research addresses the utility of pentests as a valuable Business Intelligence method, pentest outputs are not standardized and require time-consuming generation and ingestion before they can provide inputs to Business Intelligence Systems. In this study, we examined several pentest reports and propose a standardized schema for output generation in a common format that is easy for Business Intelligence Systems to ingest. We propose that this improvement will allow more rapid delivery of accurate risk information to executives, managers, and information technology professionals. The thirteen elements of our standardized schema represent the first attempt at building a pentest dimensional model, which could further ease the translation of cybersecurity risk information into business impact analyses and organizational risk registers.**

***Index Terms*—business intelligence; penetration test; pentest report; data driven BIS**

I. Introduction

Penetration Testing (pentesting) is the authorized intrusion into digital systems. The outputs provide a list of exploitable vulnerabilities that are present in the system(s), a vulnerability criticality assessment, and a prioritized list of recommended remediation actions [1]. Persons who perform these activities are known by a variety of names such as pentesters, ethical hackers, and white hat hackers. Organizations either do the pentesting themselves or they take help from third parties. These activities help prevent the loss of confidentiality, integrity, and/or availability for the organization's digital systems and seek to prevent attacks that could cause financial or reputational damage [2]. Though pentests are increasingly common, the translation of those findings into business intelligence platforms remains a problem. Many pentesters have difficulty conveying their results to the executives. Reporting the results can be a tedious task that requires a considerable amount of time and effort, as there is no standard framework for reporting pentest results. This can lead to omission of important attributes of the test results in the report, or an insufficient comprehensibility of technical details. This calls into question the tester's ability and knowledge and undermines the efficacy of the pentesting process itself.

For these reasons, we provide a standardized format for penetesting reports. This is designed to reduce the time and effort required to produce an intelligible and complete report. By providing a standardized schema for pentesting report outputs, we seek to support automation of these outputs for ingestion by business intelligence systems. To do this, we compare different penetesting reports and findings to extract the necessary ingredients for a standardized reporting schema.

II. Background

There are many tools and technologies–both manual and automatic–that are used in pentesting to perform reconnaissance, scanning, gaining, and maintaining access.

A. Reconnaissance

Reconnaissance provides initial information is collected as part of the observation, which helps to understand the target and define the scope of the engagement. The information includes gathering of IP addresses, mail servers, public domains, higher-level network topology, and so on. This can be deliberately shared during the contracting of the pentest (*white-box testing*), or part of the mandate defined by a statement of work to discover what is available (*black-box testing*). Tools like Maltego provide a relatively automated mechanism for reconnaisance activities.

B. Scanning

Scanning involves sending probes and saving the responses. Network monitoring tools such as Wireshark, network scanning tools such as Nmap, webserver vulnerability scanners like Nikto, web-application vulnerability scanners like OWASP Zap, and network vulnerability scanners like Nessus are popular tools for discovering common vulnerabilities. If testing a specific application, pentesters can also use automated tools to perform static or dynamic code analysis.

Static Analysis The application's code is examined at rest. Static Application Security Testing (SAST) tools use predefined rules to help detect security vulnerabilities. Example SAST tools include Bandit, SonarQube, Veracode, Flawfinder, Cppcheck, and RATS [3]. A relatively recent addition to the field of Static Analysis allows Security Vulnerability Prediction (SVP). SVP requires models to learn the rules and detect patterns within the code that could yeild security vulnerabilities [4], [5]. SVP, and the Vulunerability Prediciton

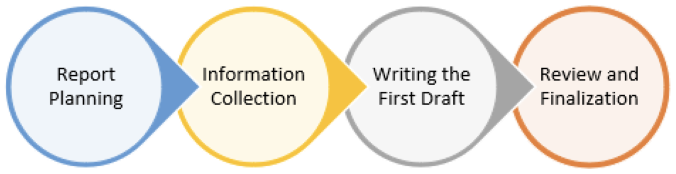

Fig. 1: Phases of Report Writing

Models they are built upon, are an active area of inquiry and have well-documented challenges to integration in automated scanning tools [6].

Dynamic Analysis The application's code is examined while running. Both commercial and open source Dynamic Application Security Testing (DAST) can be useful when the source code is unavailable and, unlike SAST tools, are not designed for any specific programming language. Example DAST tools include Burp Suite, InsightAppSec, and AppScan. Automatable DAST techiques include Web Application Security Testing, Security API Scanning, and Behaviour Driven Security Testing [7].

C. Gaining and Maintaining Access

To gain access, pentesters use information from previous stages to gain access to the application or underlying system. To maintain access, they will deploy various types of tools such as Remote Access Trojans, covert channels, and/or backdoors to escalate privileges and move laterally through a digital network. Though Gaining Access and Maintaining Access are considered two separate steps of a pentest [8], common frameworks such as Metasploit, Powershell Empire, and Cobalt Strike integrate tools that help pentesters both gain and maintain access. Some–such as Caldera–specifically seek to replicate the known tactics, techniques, and procedures of advanced persistent threats.

Generally, these tools find and exploit vulnerabilities. Applying the findings to the organization's business context and providing actionable insights are beyond the scope of currently available tools.

Once the pen testers are through with scrutinizing the weaknesses of the system, all the details have to be combined and presented in a report. Penetration Test Reports are a vital part of the entire penetration testing process since it is the proof of the test results which are presented to the company's Executives and Information Security team for them to take action according to the findings. For example, which system needs to be patched or which system needs to be upgraded, how to take further action in terms of increasing the security, and how to plan the security budget.

As early as 2010, Alharbi formalized through the SANS institute an industry standard for the phases of effective penetration test report writing [9]; this is reflected in Figure 1:

1) Report Planning – This is the first phase of report creation. In this, the testers will focus on the Report Objective and the timeline and will consider the target people for whom the report is being generated.
2) Information Gathering – In this phase the tester uses the collected data that he gathered during each step of penetration testing. The data includes the notes that he took during each phase of testing, the important screenshots which he took while running the tools, the log files of the activities, etc.
3) Writing First Draft – According to Alharbi, this step is highly recommended since it will give the tester a good idea what all things to mention in the report [9]. Also, 60% of the tester's time will be utilized in creating the first draft.
4) Review and Finalization – The first draft should be peer-reviewed before finalizing it. All the team members should participate in this process. Having other opinions could definitely be beneficial for the penetration tester as it will enhance the quality of the report and will cover all the edge cases which the tester might have left. This step is also beneficial for a situation where a team of testers were involved in the same project. Once the reviews have been collected from the peers, the report should be finalized and should be made ready to send to the clients.

III. Related Work

As early as 2010, authors such as Alharbi called for improvement in pentesting report schemas because their disparate report formats led to challenges in interpreting results [9]. More recently – in 2019 – Zakaria et al. compared eight penetration testing reports that they found online [10]. They found some of the reports missed key information, and other reports included much extraneous information. These findings led the authors to call for the community to develop a common schema.

In 2021, Alghamdi considered 20 pentest reports and described techniques of effective report writing, which provides some input to the minimum essential elements for pentest reports [11]. In the same year, Barik et al. analyzed several penetration testing tools, found disparate output formats, and proposed a standardized format for penetration testing reporting [12].

In 2022, Reddy and Pelletier considered five historical breach cases and found that the pentest output data might have helped businesses better manage risk decisions and ultimately avoid losses [13]. Though there has been a relatively robust call for pentest report output standardization, as well as a recent recognition that pentest outputs provide useful business intelligence system inputs, to date no authors have explicitly considered how a schema might allow ingestion to Business Intelligence Systems.

IV. Schema Design

For creating the standardized schema of the penetration testing report, we employed the Waterfall Design Model [14].

The remainder of this section describes our findings and analysis while following this design process, organized according to the *a.* Research, *b.* Prototyping, *c.* Coding, *d.* Testing, and *e.* Integration phases.

A. Step 1: Research

In Step 1, we continued research to examine best practices in pentest reporting outputs and considered three tools to generate outputs for our prototype schema.

We gathered six new pentest reports, which had not previously been analyzed for format and output content. Four reports came from the Collegiate Penetration Testing Competition (CPTC) – a global penetration testing competition that simulates real-world pentests [15], [16]. CPTC competitors begin at eight regions across the globe and winners from the regional level ultimately compete in a final championship round. The CPTC organization hosts a GitHub repository containing final participant team reports, which we included in our inquiry as exemplars of world-class pentest reports [17], [18], [19], [19]. Two reports came from professional pentesters. One was from Red Siege Information Security [20], a cyber security consulting firm that provides security solutions to clients. Another came from a professional example and included a video instruction on report formatting [21].

We then used the Penetration Testing Execution Standards (PTES) baseline guidance [22] to inform the necessary components from each of the six reports we selected. According to PTES, the reports should be broken down into two major sections, each meant for a different audience: Executive and Technical.

The **Executive-level** summary should contain high-level information about the test. It includes a *high-level summary*, *findings overview*, *risk severity*, and *strategic recommendations*. The target readers are upper-level management, who are also likely to be engaged in strategic and operational level risk management practices. These data are most likely to inform the Business Impact Analysis and the Organizational Risk Register. This section is written for senior managers, who are the most frequent consumers of Business Intelligence System information [23].

The **Technical-level** detail should contain all the in-depth information of the test in a way that informs the actual risk-management practices at the tactical and operational levels. It includes an *introduction/summary*, the *methodology* for the pentest, *findings*, additional context as *informational findings*, a *conclusion*, and detailed *appendices* containing scan results and additional/non-critical evidence. The findings section is arguably the most important component of the report and it contains six key items:

1) Threat Name/Severity Level: The name of of the vulnerability/exploit and its severity level (i.e. critical, high, medium, or low).
2) Description: Explains what the vulnerability and exploit are.
3) Affected Assets: The machines in the environment affected by vulnerability/exploit.
4) Recommendation/remediation: Recommended technical solution to remediate the risk/threat.
5) References: An external validation of the finding, usually a listing in a public vulnerability or exploit database, and further information on remediation.
6) Business Impact: A broader picture of the effect of these risks should be mentioned in this section.

This detail allows Information Technology professionals to understand the root of the problem and fix it. Though this level of detail is unlikely to make it to board-level products such as the Organizational Risk Register, granular data in this section allow for a more data-driven Business Intelligence System, which is important in increasingly automated sectors such as logistics [24].

TABLE I: Executive-level report components

Executive Summary Sections	PTES [14]	IT Pro TV [18]	Red Siege Information Security [19]	CPTC A [20]	CPTC C [21]	CPTC F [22]	CPTC O [23]
Summary	✓	✓	✓	✓	✓	✓	✓
Finding Overview/ Scope	✓	✓	✓	✓	✓	✓	✓
Risk Severity Representation	✓	✓	✓	✓	✓	✓	✓
Strategic Recommendation	✓	✓	✓	✓	✓	✓	✓
Roadmap	✓					✓	

TABLE II: Technical-level report components

Technical Report Sections	PTES [14]	IT Pro TV [18]	Red Siege Information Security [19]	CPTC A [20]	CPTC C [21]	CPTC F [22]	CPTC O [23]
Introduction/ Summary	✓	✓	✓	✓	✓		
Methodology	✓	✓	✓	✓		✓	✓
Findings							
1. Threat Name and Severity Level	✓	✓	✓	✓	✓	✓	✓
2. Description	✓	✓	✓	✓	✓	✓	✓
3. Affected Assets	✓	✓	✓	✓	✓	✓	✓
4. Recommendations/ Remediation	✓		✓	✓	✓	✓	✓
5. References	✓		✓	✓			✓
6. Business Impact	✓			✓	✓		✓
Informational Findings	✓			✓	✓		✓
Conclusion	✓				✓	✓	✓
Appendices	✓		✓	✓	✓	✓	✓

Tables I and II depict a comprehensive comparison of the papers according to these components. Of note, the orange-colored fields are addressed by this project through a standardized JSON schema, which is described in later steps. These orange-colored fields directly inform what might be generated by automated tools and therefore should be considered in a prototype automation schema for Business Intelligence System ingestion. In the next step, we describe this data generation and prototyping.

B. Step 2: Prototype

To generate data for our prototype schema, we considered outputs from three of the most popular [25] pentesting tools – Nmap, Nikto, and OWASP ZAP – which we described above in Section II-B. We applied those tools to exhaustively search ports and find vulnerabilites present in an instance of the DVWA (Damn Vulnerable Web Application), which is a PHP and MySQL implemented web application that is vulnerable by nature. DVWA is commonly used by the security community to research, learn and practice penetration testing tools [26]. In this case, it provided a useful set of output data during prototyping.

Each of the common tools – Nmap, Nikto, and OWASP Zap – offer a variety of output formats including XML, JSON, HTML, and text files (among others). We found that they did not, however, all offer the same standardized output:

- Nmap ⇒ XML
- Nikto ⇒ XML
- OWASP ZAP ⇒ JSON

Extracting the relevant data from these tools across multiple formats can be very time consuming. Further, for presentation to a Business Intelligence System (BIS), it is useful to have this schema output to a standardized input datatype because each entry corresponds to risk intelligence associated with each of the organizations' computer devices and the data those devices store and process. Furthermore, Yusof et al. found there is an overwhelming data integration issue across BIS that inhibits data-driven decision making [27]. By standardizing these datatypes, our prototype allows data-driven decision making associated with information technology and cybersecurity risk.

In the design selection for our schema, we considered machine-readability as a critical path requirement because our intention was to automate pentest outputs for automatic BIS ingestion. Both XML and JSON are valid options for automatic machine reading. The other main design preference was for human readability, which is likely to positively impact technology acceptance especially among senior managers. We also considered speed of ingestion and processing as a secondary design goal.

Among these output formats, JSON contains a more predictable and ultimately more human-usable format than XML [28]. Also, JSON is faster at data exchange than XML, requires fewer bytes in transit, and is supported by many data visualization tools [29]. Furthermore, in 2022, Yusof et al. specifically recommended a JSON-based standardized format as superior to XML for the overwhelming majority of Business Intelligence Systems [27]. Therefore, our prototype schema utilizes a standardized JSON output.

The variables used in our standardized JSON schema – depicted in Figure 2 – each reference a different *dimension* of the penetration test intelligence outputs, as in:

- address – URL
- host - IP address
- port - port number
- quick scan - the scans which provide the layout of the environment - for example, Nikto scans
- id - to differentiate vulnerabilities found
- method - the method used like 'GET' or 'POST'
- alert name - the vulnerability found
- risk_level - critical, high, medium, or low
- instance_count - the number of instances on which this vulnerability has been found on a given particular website
- solution - the remediation
- references - the proof of successful exploit
- affected scope - the IP addresses which are affected by this vulnerability
- other information – in this the risks which do not find any severity level, i.e., less than Low are included

This schema is noninclusive and provides a basic format and structure that can be modified further to accommodate output from additional tools or specific BIS integration requirements. In particular, automated vulnerability scanners such as Nessus provide a structured format that could be converted to this schema. Such an integration from a vulnerability scanner would lack the detail about the `references` that prove compromise exploited systems, but would still generally conform with the other data types that our schema integrates.

C. Step 3: Code

This step required transformation of pentest tool outputs into the standardized schema described above. This required two main steps:

1) **Converting** the sample XML outputs into JSON using a publicly-available tool [30].
2) **Parsing** the tool-generated reports to compile outputs into the new schema using a custom python script, depicted in Figure 3.

D. Step 4: Test

The resultant JSON file integrates the test outputs from Nmap, Nikto, and OWASP Zap, which reflects a functional proof of concept for the novel schema.

E. Step 5: Integrate

Once the schema file is created, it can be fed into any software or application programming interface that will accept data formatted in JSON.

Our proof of concept integration of standardized JSON format visualizes the data using python visualization tool libraries depicted in Figure 4. Though our proof of concept visualization is relatively simplistic, the JSON format allows for multi-dimensional modeling associated with each of the schema variables. In particular, many BIS such as Microsoft's Power BI[1] and Oracle Analytics Database[2] allow for JSON ingestion and follow-on correlation with other elements of BI.

This is useful because recent work by Chen and Siau has proven that integrating Information Technology infrastructure

[1] https://learn.microsoft.com/en-us/ [2] https://docs.oracle.com/en/

```
{
 "address": "",
 "host": "",
 "port": "",

 "open ports":[{"port number": "<port>", "service running":"<service>"}],
 "closed ports": [],
 "filtered ports": [{"port number": "<port>", "service running":"<service>"}],

 "os information": "",

 "quick scan": [{"id":"<id>","method":"<>","description":"<>"}]

 "Detailed scan": [{"id":"<id>", "alert_name":"<>",
 "description":"<>", "risk_level":"<>", "instance_count":"<>", "solution":"<>",
 "reference":"<>", "affected scope":"<>", "other information": "<>"}]
}
```

Fig. 2: Standardized JSON schema

```
import json

myjsonfile = open('nmap_basic.json', 'r')
jsondata = myjsonfile.read()
obj = json.loads(jsondata)
abc={}
abc['address'] = obj['host']['address']['@addr']
abc['host'] = obj['host']['hostnames']['hostname']['@name']
abc['ports'] = {}
abc['ports'] ['open_ports_count'] = int(len(obj['host']['ports']['port']))
abc['ports'] ['close_ports_count'] = int(obj['host']['ports']['extraports']['@count'])
abc['ports'] ['Total_ports_count'] = int(obj['scaninfo']['@numservices'])
abc['open_ports'] = []
for i in range(len(obj['host']['ports']['port'])):
           open_ports = {}
           open_ports['protocol'] = obj['host']['ports']['port'][i]['@protocol']
           open_ports['portid'] = obj['host']['ports']['port'][i]['@portid']
           open_ports['service'] = obj['host']['ports']['port'][i]['service']['@name']
           abc['open_ports'].append(open_ports)

myjsonfile2 = open('nmap_os_version.json', 'r')
jsondata2 = myjsonfile2.read()
obj2 = json.loads(jsondata2)
pqr={}
pqr['os'] = obj2['host']['os']['osmatch']['@name']

# new = {**abc, **pqr}

# print(new)
# object2 = json.dumps(pqr, indent=4)
```

Fig. 3: After XML ⇒ JSON conversion, this parser extracts each information dimension to populate the schema

with BIS positively correlates with organizational agility [31]. An important use case includes the security properties of the information technology systems used to collect, store, analyze, and report business intelligence itself. By integrating pentest output for each device, decision makers will gain a useful data source that would apply a form of meta-cognition to the BIS inputs themselves. For example, if key business decisions rely on systems or data lakes that are vulnerable to data integrity attacks, decision makers should know this and might seek additional confirmatory signals before taking a decision. Some additional integration potentials include compliance tracking, business impact analyses refinement, and private equity portfolio health.

V. Conclusions and Discussion

Business Intelligence Systems (BIS) enable data-driven insights. However, these insights often lack the cybersecurity risks that penetration tests (pentests) provide. The pentest method provides useful business risk data and this study demonstrates a novel implementation of those outputs to inform BIS. To do this, we analyzed six pentest reports to de-

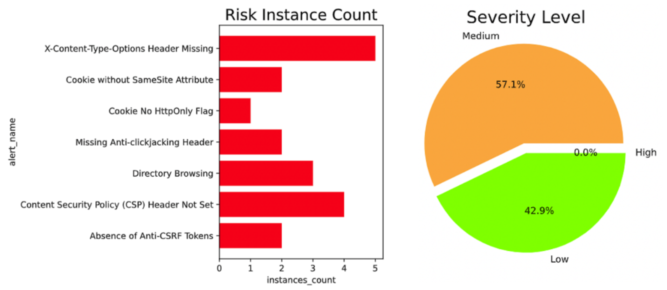

Fig. 4: Proof-of-concept visualization

termine common dimensions and opportunities for automation, developed a prototype schema to display those dimensional data, generated test output from three common pentest tools, converted those outputs into the schema, and demonstrated a proof-of-concept using basic visualization tools.

This standardized schema will help pentesters process and present the results in a rapid and extensible manner. Most importantly, this standardized schema represents the first proof of concept for integrating automatic pentest tool outputs into a format that is useful for data-driven BIS. Finally, our novel schema reduces pentest outputs into a basis spanning thirteen data elements. This represents the first step toward a usable pentest dimensional model.

VI. Limitations and Future Work

This project deals with just three tools – Nmap, Nikto, and OWASP ZAP – and there are many tools present in the market for performing a penetration test. Hence, there might be a case where the proposed JSON standardized format is missing some of the attributes which should be addressed. Therefore, future work can should consider how other vulnerability scanners and pentesting tools could enrich the format and content of this schema.

Also, this project did not include input from actual BIS users. Gathering feedback about the usability of our schema from the people who will use it will increase the practicality of this project. Furthermore, this feedback might inform dimensionality reduction of our schema to make it more efficient and practical. Hence, follow-on qualitative research should consider the ground-truth of integration of these data into BIS as well as the actual usefulness of each pentest report output field amid the myriad data-driven decisions that managers face.

VII. Acknowledgements

This research was funded in part by ongoing activities in the Eaton Cybersecurity SAFE lab at Rochester Institute of Technology's ESL Global Cybersecurity Institute. S. Sharma would like to express gratitude for the encouragement from her friends and family during this work. J.M. Pelletier would also like to acknowledge the ongoing support he receives from the *Ordo Praedicatorum*.

References

[1] P. Vats, M. Mandot, and A. Gosain, "A comprehensive literature review of penetration testing & its applications," in *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*. IEEE, 2020, pp. 674–680.

[2] C. Scully and P. Wang, "Router security penetration testing in a virtual environment," in *Information Technology-New Generations: 14th International Conference on Information Technology*. Springer, 2018, pp. 119–124.

[3] R. Croft, D. Newlands, Z. Chen, and M. A. Babar, "An empirical study of rule-based and learning-based approaches for static application security testing," in *Proceedings of the 15th ACM/IEEE international symposium on empirical software engineering and measurement (ESEM)*, 2021, pp. 1–12.

[4] N. Munaiah and A. Meneely, "Data-driven insights from vulnerability discovery metrics," in *2019 IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE)*. IEEE, 2019, pp. 1–7.

[5] H. Hanif, M. H. N. M. Nasir, M. F. Ab Razak, A. Firdaus, and N. B. Anuar, "The rise of software vulnerability: Taxonomy of software vulnerabilities detection and machine learning approaches," *Journal of Network and Computer Applications*, vol. 179, p. 103009, 2021.

[6] P. Morrison, K. Herzig, B. Murphy, and L. Williams, "Challenges with applying vulnerability prediction models," in *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, 2015, pp. 1–9.

[7] T. Rangnau, R. v. Buijtenen, F. Fransen, and F. Turkmen, "Continuous security testing: A case study on integrating dynamic security testing tools in ci/cd pipelines," in *2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE, 2020, pp. 145–154.

[8] R. Pandey, V. Jyothindar, and U. K. Chopra, "Vulnerability assessment and penetration testing: a portable solution implementation," in *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE, 2020, pp. 398–402.

[9] M. Alharbi, "Writing a penetration testing report," United States: SANS Institute., 2010.

[10] M. N. Zakaria, P. A. Phin, N. Mohmad, S. A. Ismail, M. N. Kama, and O. Yusop, "A review of standardization for penetration testing reports and documents," in *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*. IEEE, 2019, pp. 1–5.

[11] A. A. Alghamdi, "Effective penetration testing report writing," in *2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. IEEE, 2021, pp. 1–5.

[12] K. Barik, A. Abirami, S. Das, K. Konar, and A. Banerjee, "Penetration testing analysis with standardized report generation," in *3rd International Conference on Integrated Intelligent Computing Communication & Security (ICIIC 2021)*. Atlantis Press, 2021, pp. 365–372.

[13] P. S. Reddy and J. Pelletier, "The pentest method for business intelligence," in *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 2022, pp. 1117–1125.

[14] A. Sinha and P. Das, "Agile methodology vs. traditional waterfall sdlc: A case study on quality assurance process in software industry," in *2021 5th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*. IEEE, 2021, pp. 1–4.

[15] B. Stackpole and D. Johnson, "Cptc-a security competition unlike any other," 2019.

[16] N. Munaiah, A. Rahman, J. Pelletier, L. Williams, and A. Meneely, "Characterizing attacker behavior in a cybersecurity penetration testing competition," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2019, pp. 1–6.

[17] https://github.com/globalcptc/report_examples/blob/master/2020/Finals-A-reportredacted.pdf.

[18] https://github.com/globalcptc/report_examples/blob/master/2020/Finals-C-report-redacted.pdf.

[19] https://github.com/globalcptc/report_examples/blob/master/2020/Finals-O-report-redacted.pdf.

[20] https://github.com/juliocesarfort/public-pentesting-reports/tree/master/RedSiege.

[21] https://go.itpro.tv/pentest-report.

[22] multiple, "Penetration testing execution standard," 2012. [Online]. Available: http://www.pentest-standard.org/index.php/Main_Page

[23] I. Constantiou, A. Shollo, and M. T. Vendelø, "Mobilizing intuitive judgement during organizational decision making: When business intelligence is not the only thing that matters," *Decision Support Systems*, vol. 121, pp. 51–61, 2019.

[24] C. Zhou, A. Stephen, X. Cao, and S. Wang, "A data-driven business intelligence system for large-scale semi-automated logistics facilities," *International Journal of Production Research*, vol. 59, no. 8, pp. 2250–2268, 2021.

[25] M. Shah, S. Ahmed, K. Saeed, M. Junaid, H. Khan *et al.*, "Penetration testing active reconnaissance phase–optimized port scanning with nmap tool," in *2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2019, pp. 1–6.

[26] S. K. Oh, N. Stickney, D. Hawthorne, and S. J. Matthews, "Teaching web-attacks on a raspberry pi cyber range," in *Proceedings of the 21st Annual Conference on Information Technology Education*, 2020, pp. 324–329.

[27] M. K. YUSOF, M. MAN, W. A. F. W. HAMZAH, S. SAFEI, and I. ISMAIL, "Native json model for data integration in business intelligent applications," *Journal of Theoretical and Applied Information Technology*, vol. 100, no. 18, 2022.

[28] http://www.json.org/xml.html.

[29] https://www.toptal.com/web/json-vs-xml-part1#:~:text=JSON%20is%20faster%20because%20it,more%20than%20just%20data%20interchange.

[30] https://jsonformatter.org/xml-to-json.

[31] X. Chen and K. Siau, "Business analytics/business intelligence and it infrastructure: impact on organizational agility," *Journal of Organizational and End User Computing (JOEUC)*, vol. 32, no. 4, pp. 138–161, 2020.