INTEGRATING SECURITY COMPONENTS: A DEMONSTRATION, LITERATURE REVIEW, AND BENCHMARK STUDY OF SQL TOOLS IN WEB SECURITY

Author: Roberto F.

ABSTRACT

SQL injection poses a significant threat to web applications that use SQL for data storage and retrieval, allowing attackers to execute unauthorized commands against databases. This demonstration showcases the use of SQL Map along with other tools to access databases of vulnerable web applications, emphasizing the exposure of confidential information. SQL Map, an open-source tool, automates the detection and exploitation of SQL injection flaws, boasting features like database fingerprinting and command execution via out-of-band connections. The literature review focuses on projects that employ SQL Map, evaluating their strategies to exploit vulnerabilities. Sourcing from Google Scholar, the review details include recent studies, analyses, and comparative studies. Lastly, the benchmark study assesses SQL Map's effectiveness in detecting and exploiting SQL injection vulnerabilities across diverse websites, aiming to provide insights into its capabilities and limitations.

1 Introduction

In this milestone, I have integrated all the components from the previous milestones. The first portion of this milestone presents a demonstration of SQL Map. I used a set of tools to access the databases of a vulnerable web application. By the end of this demonstration, the contents of the databases that display confidential information could be displayed. Next, I incorporated a literature review on projects employing SQL Map, evaluating their strategies for exploiting vulnerabilities. Sourcing from Google Scholar, the review details include recent studies, analyses, and comparative studies. Lastly, the benchmark study assesses SQL Map's effectiveness in detecting and exploiting SQL injection vulnerabilities across diverse websites, aiming to provide insights into its capabilities and limitations.

2 Demonstration

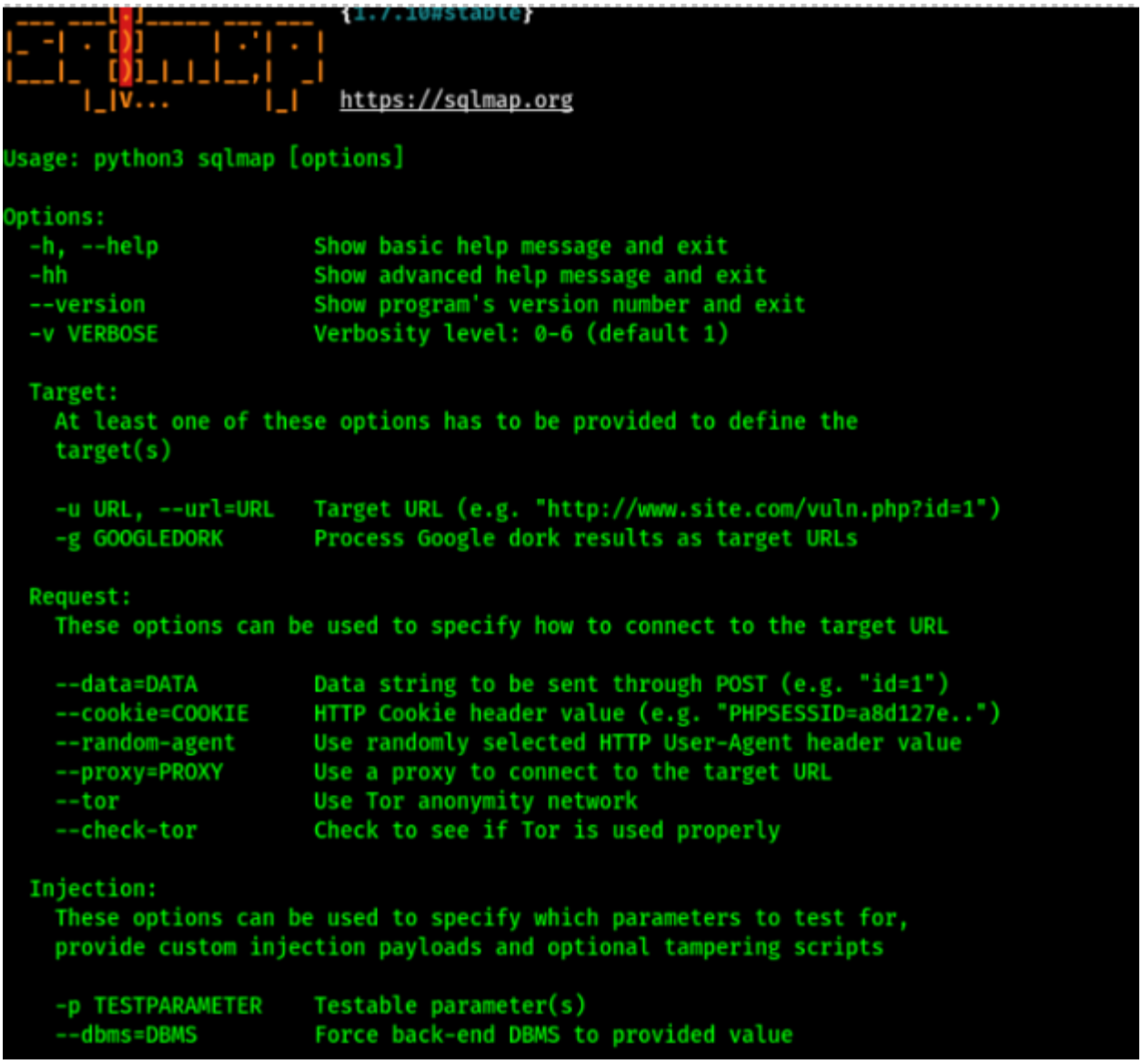
2.1 Tools:

- Kali Linux : Kali Linux is a Debian-based Linux distribution specifically designed for digital forensics and penetration testing.
- OWASP BWA : The OWASP BWA project provides a platform for practicing various penetration testing techniques, such as SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), and more
- Juice Box : Juice Shop, is an intentionally insecure web application developed by the Open Web Application Security Project. Juice Shop is used for Benchmark
- Burp Suite : Burp Suite provides a comprehensive platform for assessing web application security vulnerabilities, identifying potential security issues, and facilitating the remediation process.
- SQL Map : SQL map is an open-source tool that automates detecting and exploiting SQL injection flaws and gaining control of database servers.

2.2 SQL Map

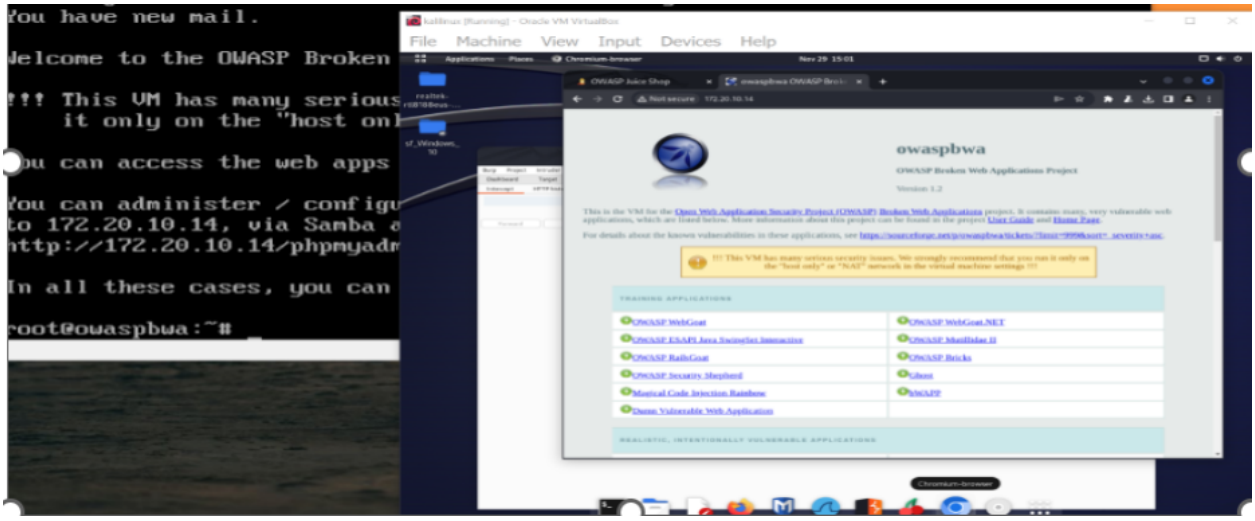
The following shows how to download the script. I am using Kali Linux as the main operating system. To demonstrate an example of SQL Map, I will use virtual machines from OWASP and Kali Linux to set up a project demonstration. This is a visual demonstration of SQL injection vulnerabilities and the utilization of SQL Map for detection and exploitation.

[1] SQL Map Installation

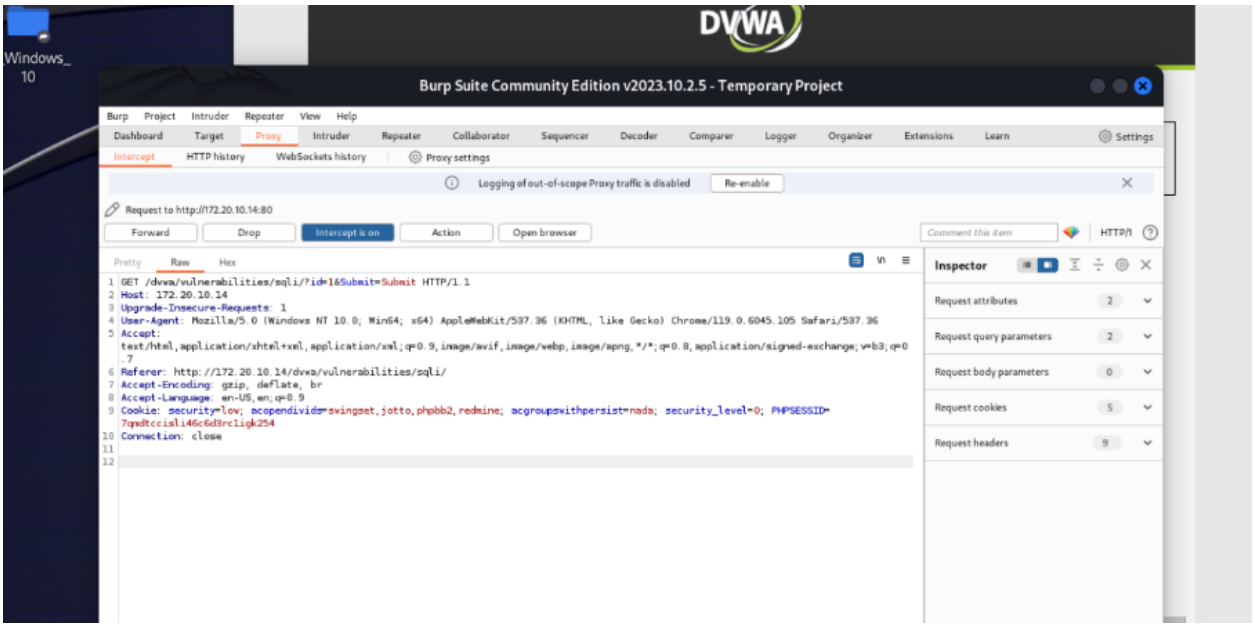


[2] SQL Map Features

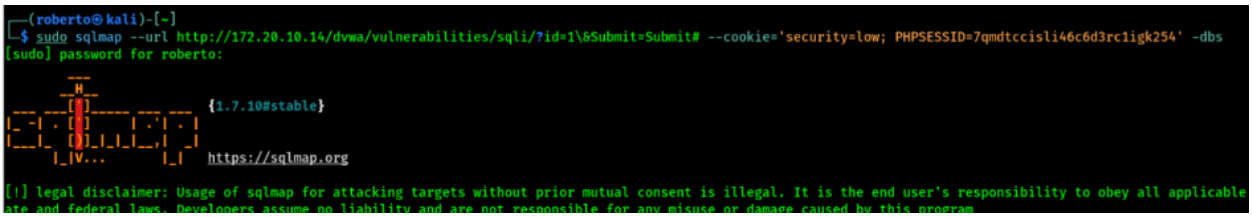
Additionally, I used the Burp Suite application integration in Kali Linux to intercept the web traffic between my web browser and the vulnerable web application and its databases. This will execute the HTTP requests and responses, which show an analysis of potential vulnerabilities. After this, the following command was run to check for possible SQL injection vulnerabilities in the web application(5).The web application databases come from OWASP Juice Shop, an intentionally insecure application designed for educational purposes. After successfully completing the command, the following vulnerable databases were found within the web application: Dvwa and Information Schema. This integration of Burp Suite and SQL Map, alongside the usage of OWASP Juice Shop's databases, is an additional step towards analyzing SQL injection vulnerabilities using SQL Map.



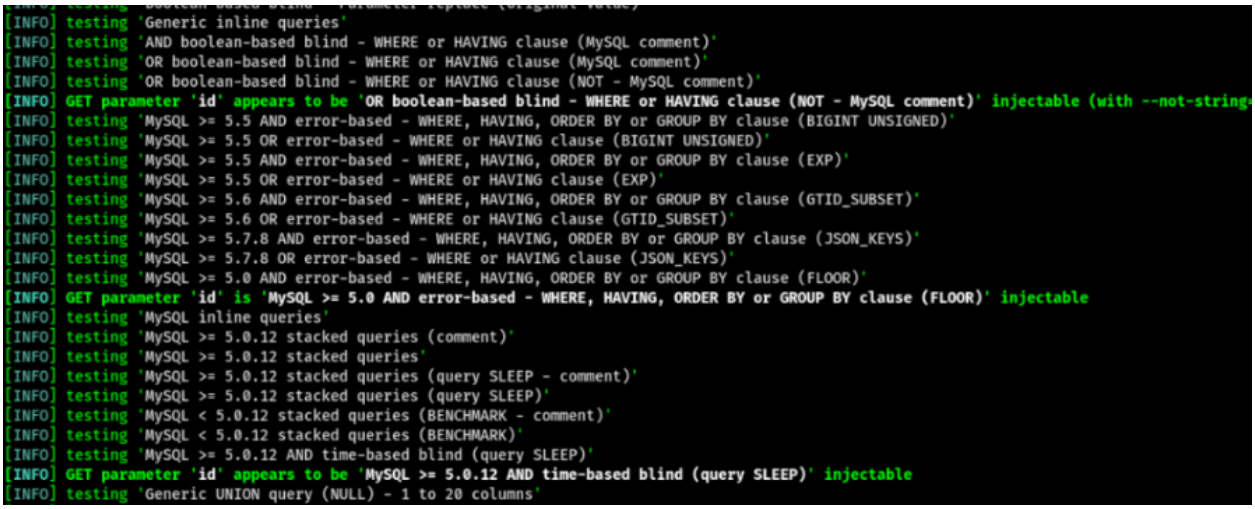
[3] OWASP Web Application



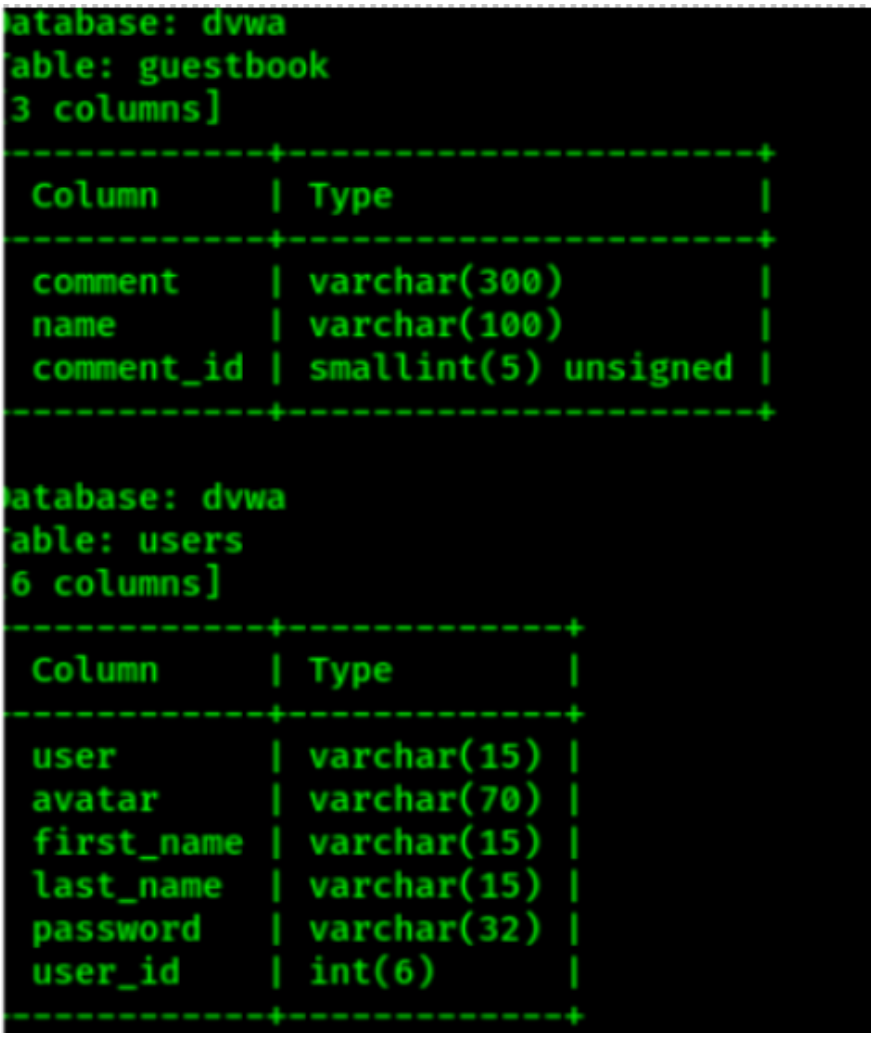
[4] Burp Suite Application For Interception



[5.1] Command To Check For Vulnerabilities and Databases



[5.2] Vulnerabilities Displayed



[5.3] Databases Displayed

The final step is to investigate these databases further, so I ran the '–tables -D dvwa' command to display the following tables shown(5.3). With further digging, the following tables are revealed from the DVWA database. The decisive step in this demonstration is to retrieve these data, as they contain user IDs, usernames, and passwords that were retrieved from the application and its database. As shown, I ran the following command to dump this information out of

the tables(6). This crucial step highlights the extent of vulnerabilities within the web application, but also emphasizes the importance of securing user information against exploitation.

[6] Command To Extract User Data From Users Database

[7] User Data Extracted

3 Literature Review

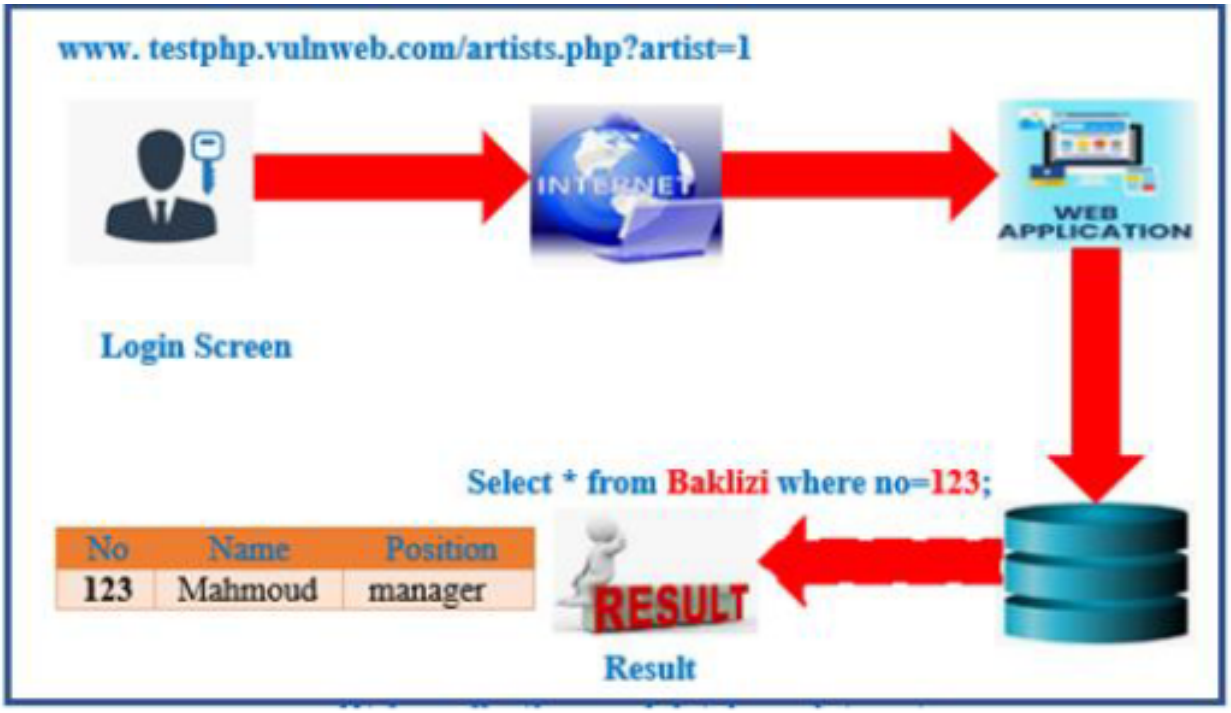
3.1 Vulnerability Analysis of Content Management Systems to SQL Injection Using SQLMAP

The study investigated SQL injection vulnerabilities across three management systems by employing Nikto and SQLMAP for penetration testing. This approach paralleled mine, utilizing SQLMAP, yet diverged in employing Nikto to broaden insights. The results demonstrated the CMS platforms' resistance to SQLi attacks while highlighting potential exploitable vulnerabilities. For example, the WordPress site, comprising 7372 scanned files, showed susceptibility to a specific attack in 16 instances, while the Drupal site, with 8217 scanned files, indicated 35 potential vulnerabilities. Despite these findings, the study advocated for proactive measures to mitigate SQL injections. In particular, SQLMAP was utilized to exploit each site through GET and POST parameters. Despite the identified vulnerabilities, the CMS platforms were shielded by configured firewalls or intrusion prevention systems (IPS/IDS), effectively thwarting SQL injection attempts.

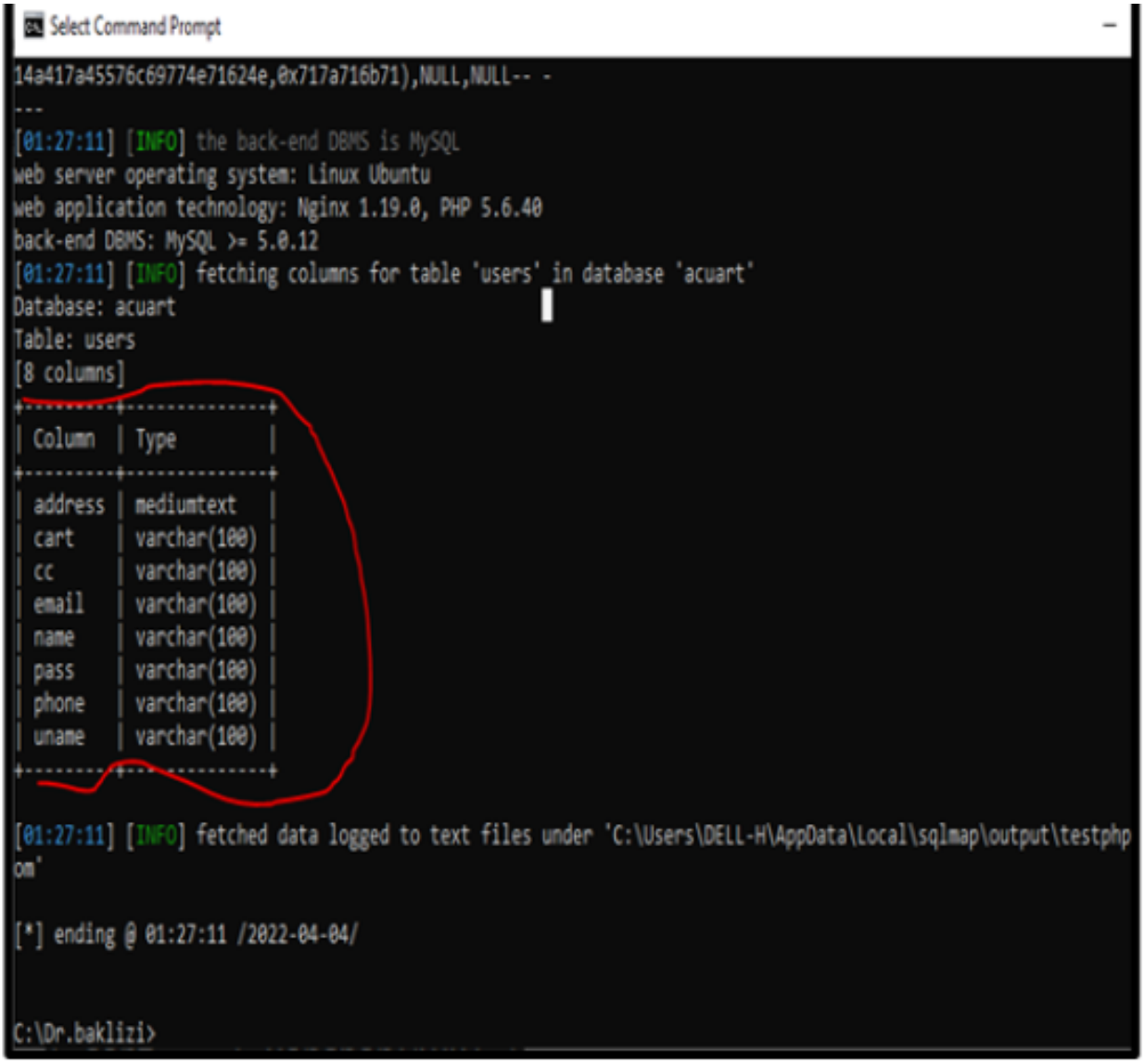
3.1.1 A Technical Review of SQL Injection Tools and Methods: A Case Study of SQL Map

The study delves into SQL injection vulnerabilities, exploring various types, detection methodologies, and tools for preemptive measures against attacks. It notably underscores the significance of SQL map and showcases its practical application on an ethical website. Through implementation, the study demonstrates successful access to the database, culminating in the extraction of both usernames and passwords.

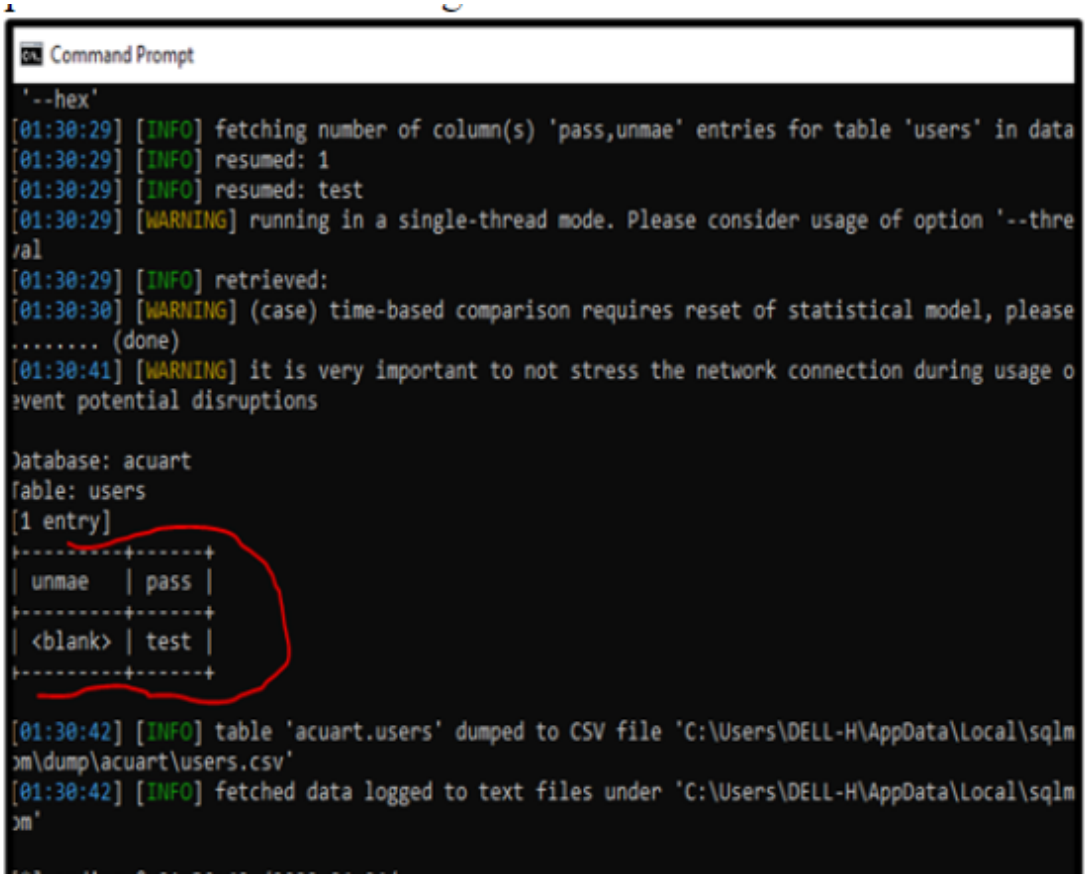
Utilizing SQLMAP within Python on a WIN10 environment, the study conducts website penetration testing. A specific command, "sqlmap.py -u http://testphp.vulnweb.com/artists.php?artist=1-DBS", is employed to extract available databases from the target site (1). Consistent with the demonstration, two databases, namely cuart and information schema, are successfully retrieved. Additionally, employing the dump command similar to the demonstration, the study exposes the username and password columns (2). In particular, the outcomes obtained are default due to the ethical nature of the targeted website.



Web Application and Database Process



[1] Database Extraction



[2] Table Contents Extraction

3.1.2 Identification and Mitigation Tool for SQL Injection Attacks (SQLIA)

The proposed methodology is focused on combating SQL injection attacks using a single tool, WebVIM. The approach, spearheaded by software testers, aims to pinpoint and address vulnerabilities within web applications. Results obtained from this endeavor underscore the tool's efficacy and precision in this domain.

Employing WebVIM, the study systematically detects SQLIA vulnerabilities, requiring user input for the URL or path to the relevant project file or web application. In particular, the tool meticulously identifies all subpages within the Web application, extracts their links, and compiles a comprehensive site map.

Furthermore, the study conducts a comparative analysis among analogous tools, with WebVIM emerging as the leader in terms of both accuracy in vulnerability identification and effectiveness in mitigation reporting.

	WebVIM	Nessus	OpenVAS	Nikto
Vulnerability Identification	Yes	Yes	Yes	Yes
Vulnerability Mitigation	Yes	No	No	No
Provides a Detailed Report	Yes	Yes	Yes	Yes

WebVIM Results on Web Applications Test

3.1.3 Research on SQL Injection Vulnerabilities and Its Detection Methods

The paper investigates the fundamentals of SQL injection attacks, along with the exploration of detection methodologies and preventive strategies, leveraging the capabilities of SQLIiscan. Throughout the study, the tool demonstrates commendable efficiency and accuracy in identifying SQL injection instances, highlighting its robustness in vulnerability detection.

Using SQLiscan, the study utilizes its capabilities to detect SQL injections. The tool operates through a crawler mechanism, which systematically retrieves URLs from web pages and feeds them into a dedicated vulnerability detection unit.

The primary functionalities of the tool encompass two key components:

- Crawler: This module specializes in URL crawling, in which it identifies the target website, initiates with an initial URL input, sends HTTP requests to the server, and subsequently analyzes the response messages.

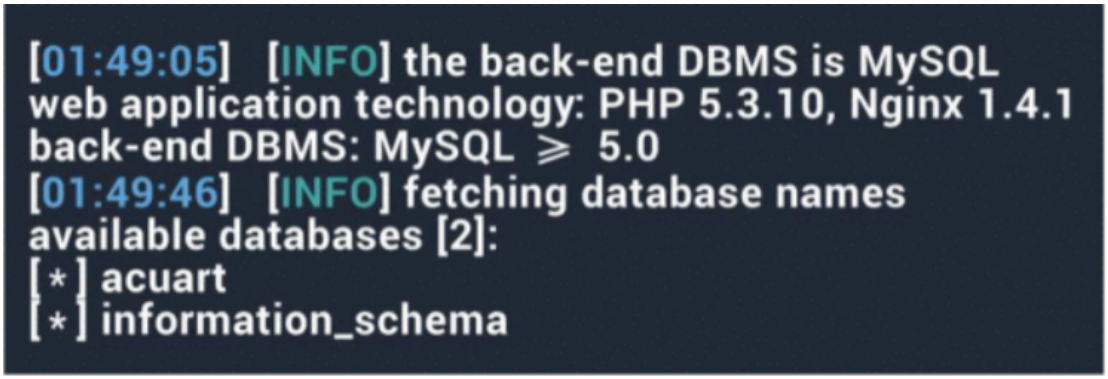
- Vulnerability Detection: Through a meticulous process, vulnerabilities are discerned by selecting target URLs, injecting preconstructed attack characters, reconstructing URLs, dispatching HTTP requests to the server, and discerning the type of request submission.

Tools	TPR-Detection	TPR	FPR-Detection	FPR
Paros	104/136	76.47%	4/10	40.00%
Wapiti	124/136	91.18%	0/10	0%
SQLiscan	128/136	94.18%	0/10	0%

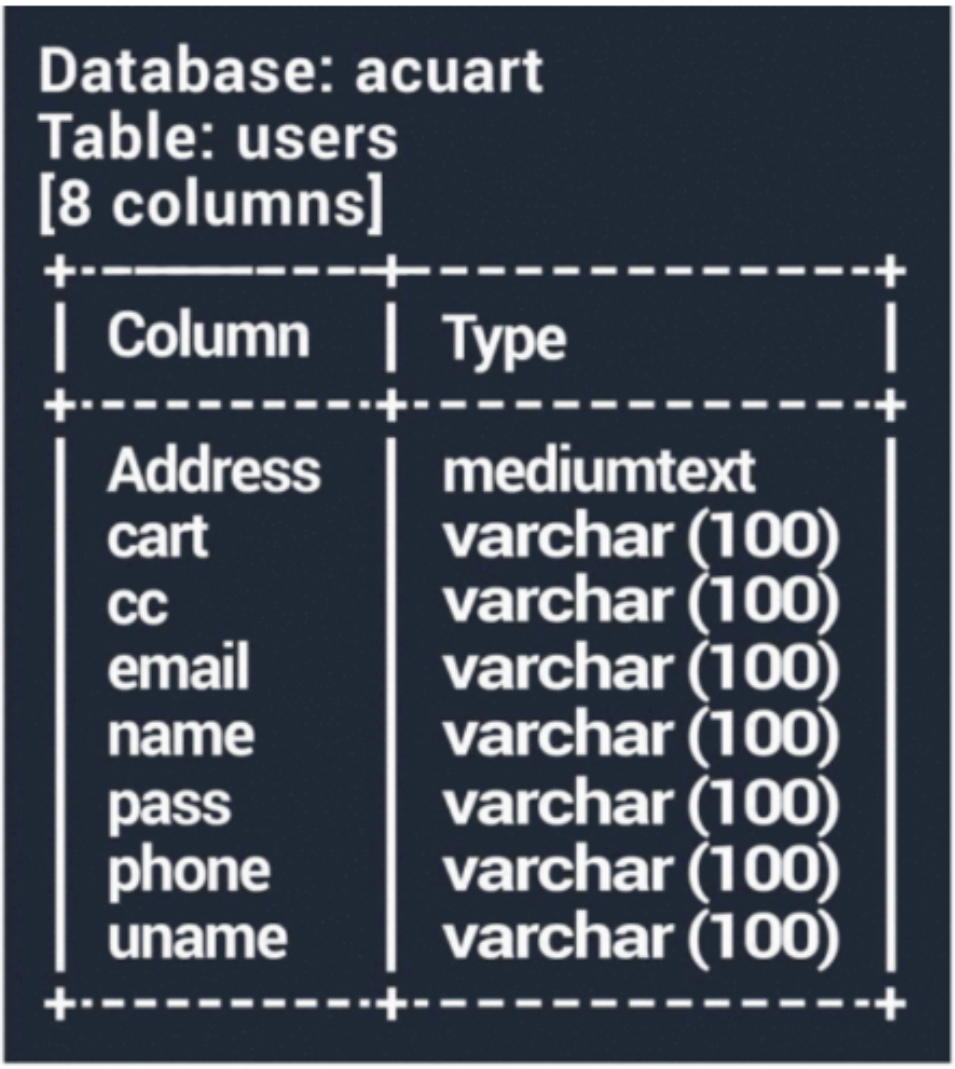
Detection Tools Findings Results

3.1.4 Overview of SQL Injection Defense Mechanisms

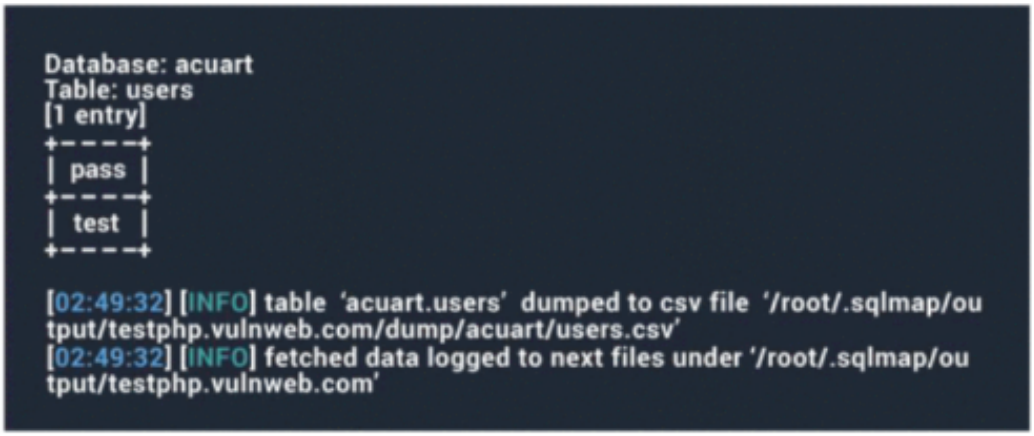
This study examines SQL injection attack methodologies while concurrently devising robust defense mechanisms for detection and prevention. Conducting the attack within a controlled environment comprising a VMware virtual setup and the Kali-Linux operating system, the study employs the SQL map tool for penetration testing. The targeted website for this demonstration is "http://testphp.vulnweb.com/listproducts.php?cat=l(1). In emulation of the demonstrated methodology, the attack proceeds with the execution of the command: "sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=1Dacuart–tables.". This directive facilitates the extraction of crucial information, such as table names and other pertinent data (2). With the requisite information gathered for subsequent exploitation, the final command is executed to extract data from the identified tables. The command utilized for this purpose is: "sqlmap -u “http://testphp.vulnweb.com/listproducts.php?cat=1” -D acuart -T users -columns" (3).



[1] Target Databases



[2] Tables Information Extracted



[3] Users Table Extracted/Credentials Revealed

3.1.5 A Detective Tool against SQL Injection Attacks Based on Static Analysis and Dynamic Monitor

The study introduces a novel approach leveraging static analysis and dynamic monitoring for the detection of SQL injection attacks, presenting a process for assessing user input risks and categorizing user behavior effectively.

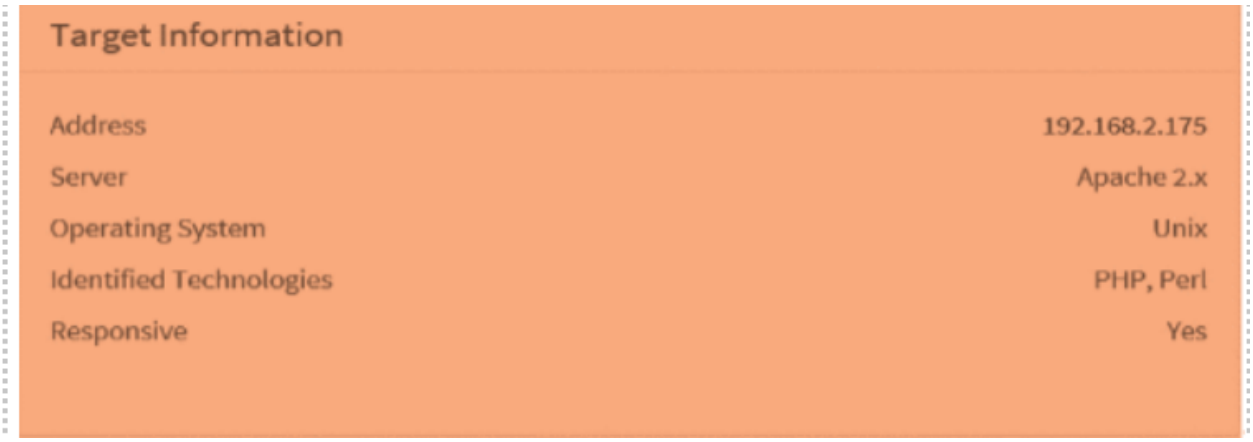
This detection tool contains several distinctive features. Firstly, it conducts an in-depth analysis of source files, culminating in the generation of comprehensive static analysis reports. Furthermore, the tool extends its capabilities to dynamically monitor all incoming server messages. Within this context, it abstracts the parameters contained in each message and subsequently substitutes undetermined elements with the identified parameters.

Position line	Query	Static Base	Danger Degree (%)	Danger Type
19	SET NAMES utf8	0	0.0	
25	SELECT * FROM users WHERE account='$account'	152	38.43	*ta i u b ti*
33	INSERT INTO users (account, password, name, IsManager) VALUES ('$account', '$password', '$name', 0)	605696	48.73	*ta i u p b ti a*
16	SELECT * FROM users WHERE account='$account' AND password='$password'	161728	49.21	*ta i u b ti*
15	UPDATE users SET IsManager=0 WHERE account='$account'	1444	42.64	*ta i u b ti*
17	DELECT FROM users WHERE account='$account'	76	37.63	*ta i u b ti*
19	SELECT * FROM users WHERE account='$account'	152	38.43	*ta i u b ti*
165	null	0	0.0	
169	null	0	0.0	
15	UPDATE users SET password='$password', name='$name' WHERE account='$account'	6145664	50.81	*ta i u b ti*
15	UPDATE users SET IsManager=1 WHERE account='$account'	1444	42.64	*ta i u b ti*
14	SELECT * FROM users	0	0.0	

Detection Tool Analysis Report

3.1.6 Automatic Protection of Web Applications Against SQL Injections: An Approach Based on Acunetix, Burp Suite and SQLMAP

The study was conducted utilizing the genuine Damn Vulnerability Application (DVWA) and simulating various SQL injection attack scenarios across different security levels: low, medium, and high. Despite the enhanced security mechanisms implemented at these levels, the results obtained showcased the same results as my demonstration. This study shares a common target as my demonstration. However, it consists of additional tools employed to exploit the application. Specifically, the Acunetix Web Vulnerability Scanner tool was deployed to automate a comprehensive vulnerability scan of the DVWA application (1). Additionally, the Burp Suite tool was utilized to intercept HTTP traffic between the browser and the DVWA application (2). Through the Acunetix Web Vulnerability Scanner, multiple vulnerabilities within the web application were identified. Subsequently, leveraging Burp Suite to intercept traffic, the results obtained revealed the presence of databases. Lastly, the tables within these databases were displayed, extracting user credentials, including usernames and passwords (3).



[1] Acunetix Web Vulnerability Scanner Results

[2] Integrating SQL Map and Burp Suite to Display Databases

[3] Format to Extract Contents From Database

3.1.7 Performance-based Comparative Analysis of Open-Source Vulnerability Testing Tools for Web Database Applications

JSQL emerges as a potent database management system, endowed with a plethora of functionalities, including database traversal, administrative page detection, file uploading, string encoding/decoding, and brute-force hashing capabilities. These attributes render it a comprehensive tool for database management tasks.

This study undertakes a comparative assessment between SQLMAP and JSQL vulnerability testing tools tailored for web database applications. This evaluation entails subjecting dummy URLs to both tools for demonstration purposes. The resulting analysis assists users in selecting the most appropriate tool based on their specific security requirements.

In this study, SQLMAP assumes the role of the primary tool for website scanning, primarily tasked with identifying SQL injection vulnerabilities, analyzing SQL-related weaknesses, and extracting sensitive information from databases. The investigation encompasses a targeted approach, outlining specific objectives and corresponding outcomes to provide a comprehensive assessment.

3.2 Methodology

Selection of websites:

- `DVWA Application`
- `Juice Box Application`
- `http://testphp.vulnweb.com/listproducts`
- `http://www.site.com/vuln.php?id=1`
- `http://testphp.vulnweb.com/listproducts.php?cat=l`
- `http://testphp.vulnweb.com/listproducts.php?cat=1Dacuart-tables`

Test parameter lists	URLNo.(s) Found Vulnerable
BIGINT:exp check:	None
Double:exp check	None
Groupby:float_req_check	1,6,7,8,10,12,14
JSON	None
XML:extract value	1,6,7,8,10,12,14
Strategy time check	3,7,13
Strategy blind check	3,7,13

Web Application List and Vulnerabilities Results

3.2.1 Beyond the Basics: A Study of Advanced Techniques for Detecting and Preventing SQL Injection Attacks

The research study critically assesses recent advancements in combating SQL injection attacks, elucidating effective models, methodologies, and strategies geared toward mitigating, detecting, and averting these prevalent risks. It insists that organizations stand to bolster their defensive capabilities against SQL injections, safeguard sensitive data, and uphold the operational integrity of their online applications by leveraging these sophisticated techniques. This research undertakes a comprehensive exploration of preventive measures and techniques pertaining to mitigating SQL injection vulnerabilities. Consequently, the study outlines detection methods ranging from signature-based to machine learning-based approaches, depending on the chosen strategy. Moreover, it underscores the pivotal role of preventive measures in fortifying system security, thereby averting the potential exploitation of SQL injection vulnerabilities.

Detection Techniques	Narrative	Merits	Demerits
Behavioral Analysis [17]	Monitoring and analyzing user behavior when interacting with a web application is a component of behavioral analysis. It searches for strange behaviors that might be signs of SQL injection attacks, such as a high volume of query inputs or odd navigation patterns.	- Recognises unusual attack patterns and query patterns. - Capable of developing assault methods. - Offers a preventative defense strategy.	- Requires accurate user behavior profiling. - Potential for false positives as a result of actual odd user behavior. - Difficult to execute and perfect.
Signature-Based [18]	The use of predetermined patterns or the signatures of well-known SQL injection attack techniques is used in signature-based detection. To identify probable SQL injection attempts, it examines incoming queries to these signatures.	- Capable of picking up well-known attack patterns. -Low rate of false positives.	- Restrictions to well-known assault motifs. - Ineffective against fresh or unique attacks. - Requires periodic modifications to the signature.
Taint Analysis [19]	To find any instances when contaminated data is utilized in SQL queries without the required sanitization, taint analysis monitors the flow of user inputs across the program. In order to find possible injection vulnerabilities, it tracks the beginning and spread of user inputs.	- Offers accurate injection point detection. - Context-sensitive vulnerability detection is effective.	- Requires access to the source code of the program. - Due to complicated data flow, it could produce false positives. - Takes a lot of time for complicated applications.
Anomaly-Based [20]	Analysis of the behavior and traits of SQL queries is used in anomaly-based detection to spot departures from typical use patterns. A baseline of valid requests is established, and any unexpected or suspect query behavior is flagged.	- Effective in spotting unknown or zero-day attacks: able to change with new attack methods. - Offers a preventative defense strategy.	- A higher rate of false positives. - It is necessary to accurately profile typical inquiry behavior. - May require a lot of resources.
Database Auditing [21]	All database operations, including SQL queries, are tracked and recorded during database auditing. These logs may be examined to find odd patterns or requests that can reveal information about possible SQL injection attacks.	- Full insight into database operations. - Offers data to support forensic inquiries. - Capable of identifying known and unidentified injection attempts.	- Needs more storage and monitoring infrastructure. - Log analysis and interpretation can be difficult. - May cause the database server to experience performance overhead.
Runtime Monitoring [22]	Monitoring SQL query behavior during application runtime is known as "runtime monitoring." It records and examines SQL statements, ensuring their integrity and picking up on any shady or malicious activity.	- Enables dynamic response and prevention of malicious queries. - Allows for real-time detection of injection threats.	- Needs more infrastructure for monitoring. - Could result in performance overhead. - Unexecuted queries cannot be used to find vulnerabilities.
Input Validation and Sanitization [23]	User inputs are strictly validated as part of input validation and sanitization to make sure they follow anticipated forms and to weed out any potentially dangerous input. Before utilizing user-supplied data in SQL queries, this approach ensures its security and integrity.	- Offers a preventative defense strategy. - Assists in completely avoiding injection vulnerabilities. - Easy to implement with little overhead.	- Depends on precise and thorough input validation procedures. - Updates can be needed to deal with changing attack methodologies. - Ineffective when defending against sophisticated injection attempts.

4 Benchmark

4.1 Introduction

SQL Map is an open-source tool that automates detecting and exploiting SQL injection flaws and gaining control of database servers. It features a powerful detection engine, niche features, and a range of switches like database fingerprinting, over-data fetching, accessing the underlying file system, and executing commands via out-of-band connections.This case study focuses on tools within the SQL space, demonstrating the project's features, and evaluating its similarities and differences. The objective of this benchmark is to investigate the similarities and differences between SQL Map and other tools in terms of their approaches to SQL injection vulnerability detection and exploitation. Compare and contrast the features and functionalities of SQL Map with other tools within the SQL space, including SQL Finder and SQL Scanner.

4.2 Methodology

Selection of websites:

- `DVWA Application`
- `Juice Box Application`
- `http://testphp.vulnweb.com/listproducts`
- `http://www.site.com/vuln.php?id=1`
- `http://testphp.vulnweb.com/listproducts.php?cat=l`
- `http://testphp.vulnweb.com/listproducts.php?cat=1Dacuart--tables`

4.3 Benchmark Setup

- Test environment: Kali Linux, SQL Map, SQL Finder and SQL Scanner.
- Description of the websites and web applications included in the study: The following are configured to have vulnerabilities and are intended to be exploited. These sources all come from other demonstrations in the literature review, and I ultimately replicated them to list my findings and compare my demonstration with using SQL Map.
- Procedure for executing vulnerability scans: As shown in my demonstration and the findings from replicating these other demonstrations, the features within SQL Map are much superior to these two development tools. I used compatible tools within my Kali Linux to then use SQL Map to exploit the web applications to access the information within the databases. Ultimately, these other comparable tools have limited features and are still in development. There are also no demonstrations since they are not as popular, but I used the same targets to find vulnerable information for exploitation.

4.4 Results and Analysis

The following SQL Map features revealed critical vulnerabilities, exposing the information of the database schema, usernames, and passwords, and posing a security risk. Additionally, it identified authentication bypass vulnerabilities, allowing me to bypass security controls and gain access to restricted areas.

Within SQL Finder, these various scans on vulnerable websites for SQL injection vulnerabilities were found to be boolean-based, error-based, time-based, and union-based. Boolean-based blinding involves sending SQL queries to the database and analyzing the application's response to determine if the injected conditions are true or false. Error-based involves injecting SQL queries that intentionally generate errors, revealing information about the database structure or contents. Time-based blinding involves sending SQL queries that cause delays in the application's response time, allowing attackers to infer information about the database.

Lastly, SQL Scanner is in its early stages of development; there are not many feats and only detection. It can be an introduction to demonstrate SQL injection, and in this case, I used vulnerable web applications that I found using SQL Map to verify if they were vulnerable and whether they were right and displayed them in the output.

When comparing the detection rates across different website categories, SQL Map consistently revealed critical vulnerabilities across all categories. These vulnerabilities included exposures to database schema information,

usernames, and passwords, posing significant security risks. Furthermore, SQL Map identified authentication bypass vulnerabilities, indicating weaknesses in security controls that could lead to unauthorized access. This same approach can be used with SQL Finder, but further exploitation to get access isn't possible. Lastly, SQL Finder can be a verification tool to make sure it's vulnerable before doing penetration testing.

4.5 Analysis of exploitation and challenges

In this benchmark of successful exploitation attempts, SQL Map proved highly effective in identifying vulnerabilities and used other tools to fully exploit web applications and retrieve user data. Since the tool is very popular and there are other demonstrations, I was able to navigate through the tool pretty easily. Furthermore, I integrated and compared other demonstrations from the literature review and was able to successfully extract the same information as in my demonstration. There are multiple approaches to exploiting the web application, but ultimately all have the same result. Next, SQL Finder was also highly effective in identifying vulnerabilities, primarily focused on detection rather than exploitation. With further development and employing additional tools, further exploits can take place after detecting a vulnerability. Lastly, the SQL Scanner is the same and will look more like a beginner tool for understanding SQL injection.

4.6 Summary of Findings

This benchmark was conducted on SQL vulnerability scanning using various tools, including SQL Map, SQL Finder, and SQL Scanner. The SQL Map scans revealed critical vulnerabilities across different website categories, exposing sensitive information such as database schema, usernames, and passwords. In addition, authentication bypass vulnerabilities were identified, highlighting weaknesses in security controls.

SQL Finder scans detected various types of SQL injection vulnerabilities, including boolean-based, error-based, time-based, and union-based injections. Each type of injection revealed different aspects of database structure or content, posing security risks to vulnerable websites. Although SQL Map proved highly effective in both identifying vulnerabilities and exploiting them to retrieve user data, SQL Finder primarily focused on detection rather than exploitation. However, with further development and integration with additional tools, SQL Finder could potentially lead to exploitation after detecting vulnerabilities. Next, SQL Scanner, in its early stages of development, primarily served as an introductory tool to demonstrate SQL injection. It lacked advanced features for exploitation, but could still verify the vulnerability of web applications found using SQL Map. Overall, the analysis highlighted the critical importance of robust security measures to mitigate SQL injection vulnerabilities and the need for comprehensive testing and detection tools to safeguard against potential exploits.

Feature	SQL Map	SQL Finder	SQL Scanner
Detection Rate	High	High	Moderate
Exploitation Success Rate	High	Low	Low
Ease of Use	Moderate - Advanced	Easy	Beginner-friendly
Focus	Both detection and exploitation	Primarily detection	Primarily detection

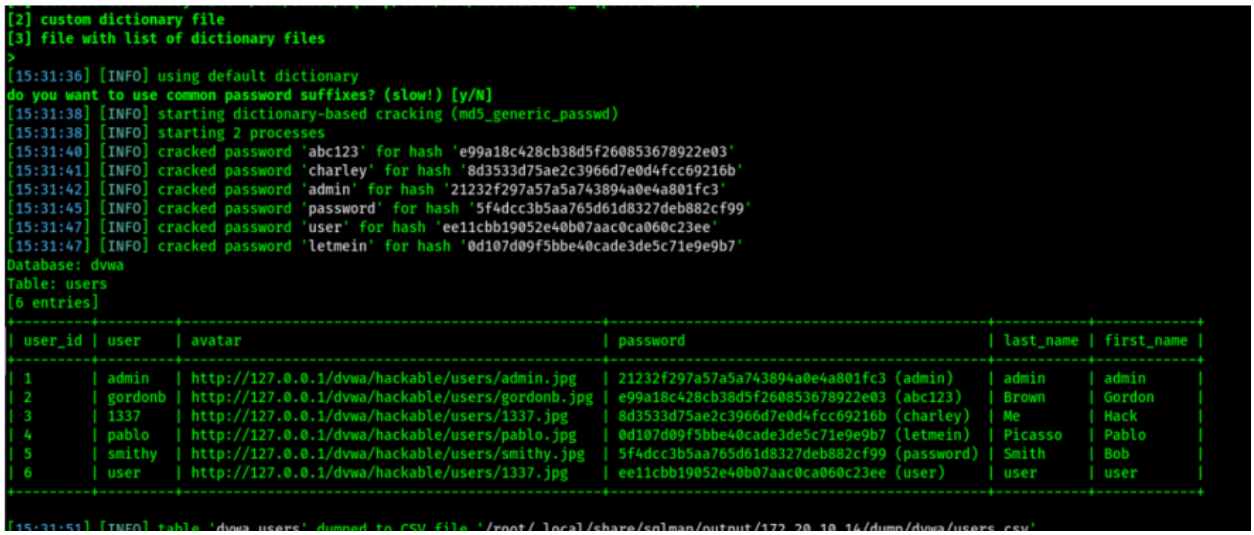
Tool Features

Tool	Boolean-Based Blind SQLi	Error-Based SQLi	Time-Based Blind SQLi	UNION-Based Query
SQL Map	High	Moderate	Moderate	High
SQL Finder	High	Moderate	Moderate	High
SQL Scanner	Low	Low	Low	Low

Web Application Results using Tools

Tool	DVWA	JUICE SHOP
SQL Map	- High detection - Gained Access to DVWA Database and retrieved user ID, usernames, and passwords.	- High detection - Gained Administrator Access - Displays registered user information
SQL Finder	- Moderate detection - No Features for Exploitation	- Moderate detection - No Features for Exploitation
SQL Scanner	- Low detection only reads that application is vulnerable	- Low detection only indicates that application is vulnerable

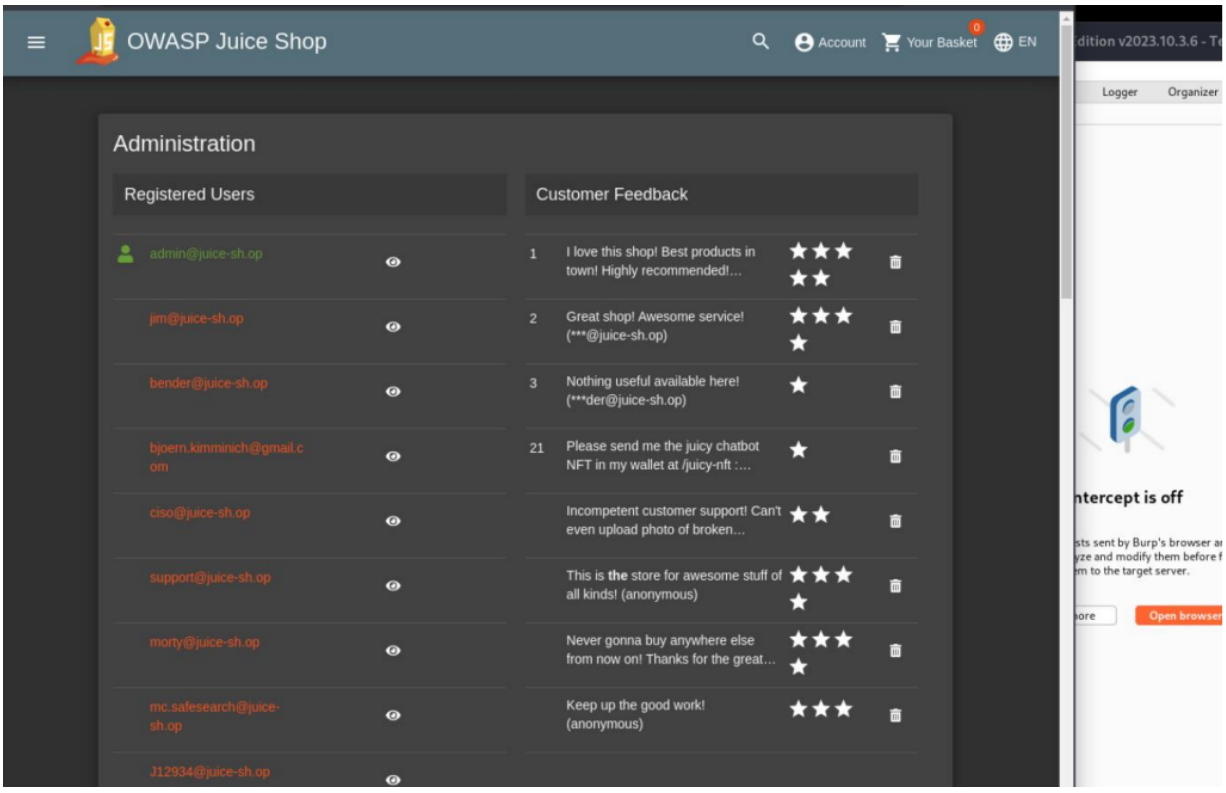
Web Application Results using Tools



```
[2] custom dictionary file
[3] file with list of dictionary files
>
[15:31:36] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]
[15:31:38] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[15:31:38] [INFO] starting 2 processes
[15:31:40] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[15:31:41] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[15:31:42] [INFO] cracked password 'admin' for hash '21232f297a57a5a743894a0e4a801fc3'
[15:31:45] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[15:31:47] [INFO] cracked password 'user' for hash 'ee11cbb19052e40b07aac0ca060c23ee'
[15:31:47] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
Database: dvwa
Table: users
[6 entries]
+---------+---------+---------------------------------------------+-----------------------------------------+-----------+------------+
| user_id | user    | avatar                                      | password                                | last_name | first_name |
+---------+---------+---------------------------------------------+-----------------------------------------+-----------+------------+
| 1       | admin   | http://127.0.0.1/dvwa/hackable/users/admin.jpg   | 21232f297a57a5a743894a0e4a801fc3 (admin)    | admin     | admin      |
| 2       | gordonb | http://127.0.0.1/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)   | Brown     | Gordon     |
| 3       | 1337    | http://127.0.0.1/dvwa/hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)  | Me        | Hack       |
| 4       | pablo   | http://127.0.0.1/dvwa/hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)  | Picasso   | Pablo      |
| 5       | smithy  | http://127.0.0.1/dvwa/hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |
| 6       | user    | http://127.0.0.1/dvwa/hackable/users/1337.jpg    | ee11cbb19052e40b07aac0ca060c23ee (user)     | user      | user       |
+---------+---------+---------------------------------------------+-----------------------------------------+-----------+------------+

[15:31:51] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/172.20.10.14/dump/dvwa/users.csv'
```

DVWA Application User Access



Juice Shop Application User Access

5 Conclusion

In this milestone, I have integrated all components from the previous milestones. The first portion of this milestone showcases a demonstration of SQL Map. I used a set of tools to access the databases of a vulnerable web application. By the end of this demonstration, I could display the contents of the databases that display confidential information. The literature review examines SQL injection vulnerabilities and detection methods, and it describes studies and tools used to identify and mitigate security risks. Tools like Nikto and SQLMap show potential exploits in management systems and how someone should follow the need for preventive measures. SQLIiscan, Web VIM, and Acunetix/Burp Suite/SQLMap are effective in detecting SQL injection attacks. This review also discusses advanced defense mechanisms against SQL injections and the importance of ongoing research to protect web applications and sensitive data. Lastly, the benchmark used SQL vulnerability scanning tools like SQL Map, SQL Finder, and SQL Scanner to identify critical vulnerabilities in web applications and website categories, including database schema, usernames, passwords, and authentication bypass vulnerabilities.

References

Ojagbule, Olajide, Hayden Wimmer, and Rami J. Haddad. "Vulnerability analysis of content management systems to SQL injection using SQLMAP." SoutheastCon 2018. IEEE, 2018.

Baklizi, M. ., I. . Atoum, N. . Abdullah, O. A. . Al-Wesabi, A. A. . Otoom, and M. A.-S. . Hasan. “A Technical Review of SQL Injection Tools and Methods: A Case Study of SQLMap”. International Journal of Intelligent Systems and Applications in Engineering, vol. 10, no. 3, Oct. 2022, pp. 75-85, https://www.ijisae.org/index.php/IJISAE/article/view/2141.

W. H. Rankothge, M. Randeniya and V. Samaranayaka, "Identification and Mitigation Tool for Sql Injection Attacks (SQLIA)," 2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS), RUPNAGAR, India, 2020, pp. 591-595, doi: 10.1109/ICIIS51140.2020.9342703.

T. Zhang and X. Guo, "Research on SQL Injection Vulnerabilities and Its Detection Methods," 2020 4th Annual International Conference on Data Science and Business Analytics (ICDSBA), Changsha, China, 2020, pp. 251-254, doi: 10.1109/ICDSBA51020.2020.00071.

I. Tasevski and K. Jakimoski, "Overview of SQL Injection Defense Mechanisms," 2020 28th Telecommunications Forum (TELFOR), Belgrade, Serbia, 2020, pp. 1-4, doi: 10.1109/TELFOR51502.2020.9306676.

Z. Liu and L. Xu, "A Detective Tool against SQL Injection Attacks Based on Static Analysis and Dynamic Monitor," 2013 10th Web Information System and Application Conference, Yangzhou, China, 2013, pp. 195-198, doi: 10.1109/WISA.2013.45

R. Bouafia, H. Benbrahim and A. Amine, "Automatic Protection of Web Applications Against SQL Injections: An Approach Based On Acunetix, Burp Suite and SQLMAP," 2023 9th International Conference on Optimization and Applications (ICOA), AbuDhabi, United Arab Emirates, 2023, pp. 1-6, doi: 10.1109/ICOA58279.2023.10308827.

A. K. Mishra and A. Kumar, "Performance-based Comparative Analysis of Open Source Vulnerability Testing Tools for Web Database Applications," 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-5, doi: 10.1109/ICCCNT49239.2020.9225324.

A. Goyal and P. Matta, "Beyond the Basics: A Study of Advanced Techniques for Detecting and Preventing SQL Injection Attacks," 2023 4th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2023, pp. 628-631, doi: 10.1109/ICOSEC58147.2023.10276077.