THREE AMIGOS TEST PLAN

COMP 4110 – Software Verification and Testing
Group 3
Team Members: Keerthana Madhavan, Charles Corro, and Het Patel

Table of Contents

Introduction

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks, and what approach will be used within the project. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

1.1 Objectives

Test Case Three Amigos is a web-based Test Management tool used to test for web application vulnerabilities and as well as display the results of running those tests. This tool is a new product that will be written in Python and Flask. The test team is responsible for testing the case study website Damn Vulnerable Web Application (DVWA) and ensuring it meets their needs. The test team is both the developer and the tester in this project.

Phase 1 of the project will deliver TCTA (Test Case Three Amigos) with functionality to test a website to identify its vulnerabilities or flaws based on the Open Web Application Security Project (OWASP) Principles. This will allow the test team to input a website URL and identify security flaws. Must have functionality is considered more important than the delivery date in this project because security problems are the root cause of damaged reputation and financial losses for most companies'

1.2 Team Members

Resource Name	Role
Keerthana Madhavan	Developer/Tester
Charles Corro	Developer/Tester
Het Patel	Developer/Tester

2 Scope

The initial phase will include all 'must have' requirements. These and any other requirements that get included must all be tested. At the end of Phase 1, a tester must be able to:

1. Input a "URL" of a website to we tested.
2. With a non-incremental approach, our tool will run and test for several cases.
3. Save it
4. Retrieve it and can then view the issue in detail
5. Mitigation Steps based on OWASP Recommendation
6. Rerun the scan

As the team works with the product, tests for flaws, it will be improved for accuracy. Load testing will not be considered part of this project since the user base is known and not an issue.

3 Assumptions / Risks

3.1 Assumptions

This section lists assumptions that are made specific to this project.

1. Delivery of the product is in a format that the user can benefit. All that Three Amigos Testing will require is for the user to enter the URL.

3.2 Risks

The following risks have been identified along with the appropriate action identified to mitigate their impact on our to be built tool. The impact) of the risk is based on how the project would be affected if the risk was triggered. The trigger is what milestone or event would cause the risk to become an issue to be dealt with.

#	Risk	Impact	Trigger	Mitigation Plan
1	Scope Creep – as testers become more familiar with the tool, they will want more functionality	High	Delays in implementation date	Each iteration, functionality will be closely monitored. Priorities will be set and discussed by users/stakeholders. Since the driver is functional and not on time, it may be necessary to push the date out.
2	Changes to the functionality may negate the tests already written and we may lose some test cases already written.	High – to schedule and quality	Loss of all test cases	Our tool needs to be loosely coupled.

4 Test Approach

The project is using an agile approach, with weekly iterations. At the end of each week, the requirements identified for that iteration will be delivered to the team and will be tested. Exploratory testing will play a large part in the testing as the team has never used this type of tool and will be learning as they go. Tests for planned functionality will be created and added to Three Amigos as we get iterations of the product.

4.1 Test Automation

Automated unit tests are part of the development process, but no automated functional tests are planned at this time.

5 Test Environment

Damn Vulnerable Web Application, XAMPP Server. This will be run on Kali Linux on a local server and the URL will be used in our tool to be tested.

1. https://dvwa.co.uk
2. https://google-gruyere.appspot.com/5264351517007722021185648214808648152 57/

6 TimeLine

6.1 Test Schedule

The initial test schedule follows:

Task Name	Start	Finish	Effort	Comments
Test Strategy	March 10th	March 13th	3 d	In progress
Code Testing Tool	March 13th	March 20th	7 d	Features will be split between team members.
Create initial test estimates	March 21st	March 22nd	1 d	Test a sample URL. Damn Vulnerable Web Application
Resolution of final defects and final build testing	March 22nd	March 23rd	1d	Find code flaws
Deploy to Staging environment	-	-	-	Host Web Tool
Performance testing	-	-	-	To be tested by other users.
Release to Production	-	-	End of Semester	Submit the project in BB

6.2 Deliverables

Deliverable	For	Date / Milestone
Test Plan	Professor/Test Team	March 9th
Test Results	Test Team	March 30th
Test Status report	QA Manager, QA Director	March 30th
Presentation	All team members	TBA

6.3 OWASP TOP 10 Web Application Security

- A01:2021-Broken Access Control
- A02:2021-Cryptographic Failures
- A03:2021-Injection
- A04:2021-Insecure Design
- A05:2021-Security Misconfiguration
- A06:2021-Vulnerable and Outdated Components A07:2021-Identification and Authentication Failures

- A08:2021-Software and Data Integrity Failures
- A09:2021-Security Logging and Monitoring Failures
- A10:2021-Server-Side Request Forgery
- A02:2021-Cryptographic Failures
- A03:2021-Injection
- A04:2021-Insecure Design
- A05:2021-Security Misconfiguration
- A06:2021-Vulnerable and Outdated Components
- A07:2021-Identification and Authentication Failures
- A08:2021-Software and Data Integrity Failures
- A09:2021-Security Logging and Monitoring Failures
- A10:2021-Server-Side Request Forgery

6.4 Our Testing Strategy

- We will be testing the following Categories in the OWASP 10. Our team members came up with this checklist. We will be using open source tools and python libraries to code our Three Amigos testing product.

Information Gathering					
☐ **WAP-INFO-001**	Check for information leakage by conducting search engine discovery and reconnaissance.	Ex. site:owasp.org cache:owasp.org Tools: Google Hacking, Shodan, Baidu, netcat	☐ Use a search engine like google chrome or safari to search for network diagrams, credentials, error message content and configuration.	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-INFO-002**	Fingerprint Web Server	nc 202.41.76.251 80 HEAD / HTTP/1.0 nc apache.example.com 80 Tools: netcat, httprecon, httprint	☐ Find the version and type of a running web server to determine known vulnerabilities and the appropriate exploits	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-INFO-003**	Review Webserver Metafiles for Information Leakage	curl -0 http://www.google.com/robots.txt	☐ *Analyze robots.txt and identify <META> Tags from website.*	☐ Yes ☐ No	☐ Passed ☐ Partially failed

					☐ failed

		Tools: Browser, curl, wget, rockspider			
☐ **WAP-INFO-004**	Map application architecture	Ex. GET /webconsole/ServerInfo.jsp%00 HTTP/1.0 Tools: Browser, curl, wget	☐ Identify the application architecture like the web languages used, application server, proxies and the database backend	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-INFO-005**	Enumerate Applications on Webserver	Ex. nmap -PN -sT sV -p0-65535 192.168.1.100 Dns: host -l www.owasp.org ns1.secure.net Tools: dnsrecon, Nmap, fierce, Recon-ng	☐ Find web applications on the server including virtual hosts and subdomains, ports and dns	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-INFO-006**	Fingerprint Web application Framework	nc 127.0.0.1 80 Tools: netcat, Whatweb	☐ Find the type of web application framework/CMS from HTTP headers, Cookies, Source code, Specific files and folders.	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-INFO-007**	Fingerprint Web application	Cookies – GET / HTTP/1.1 Nc 127.0.0.1 80 Python BlindElephant.py	☐ Find the type of web application framework from HTTP headers, cookies, source code, specific files and the folders.	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed

		http://website drupal Tools: Wappalyer, Whatweb, BlindElephant, netcat			

AUTHENTICATION

☐ **WAP-AUTH-001**	Testing for default credentials	Try default usernames such as: admin, administrator, root, system, guest, operator, superuser Nikto -h <Hostname/IP> -id <id:pass> or <id:pass:realm> Tools: Burp Proxy, ZAP, nikto	☐ Testing for credentials of common applications, Testing for default password for new accounts	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-AUTH-002**	Testing for Weak password policy	hydra localhost mysql -l root -p rootpass hydra localhost mysql -L logins.txt P password.txt Tools: Burp Proxy, ZAP, Hydra	☐ Find out the resistance of the application against brute force password guessing.	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed

☐ **WAP-AUTH-003**	Testing for weak password change or reset functionalities	Tools: Browser, Burp Proxy, ZAP	☐ Test the password reset function and test password change whether to have password in plain test, send via email or CSRF vulnerability	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-AUTH-004**	Testing for Credentials		☐ Check referrer whether its HTTP or HTTPs.	☐ Yes ☐ No	☐ Passed
	Transported over an Encrypted	Tools: Burp Proxy, ZAP	Sending data through HTTP and HTTPS.		☐ Partially failed ☐ failed
☐ **WAP-AUTH-005**	Testing for Weaker authentication in alternative channel	Sample example: http://example.com For authentication: http://example.com/myaccount/ Tools: Browser	☐ Understand the primary mechanism and Identify other channels (Mobile App, Call center, SSO)	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed
☐ **WAP-AUTH-006**	Test for Privilege Escalation	Tools: ZAP	☐ Test for role/privilege manipulate the values of hidden variables.	☐ Yes ☐ No	☐ Passed ☐ Partially failed ☐ failed