**WEB APPLICATION PENETRATION TEST REPORT**

**PHASE 4**

**ADEWUSI OLUWASEYI**

**WRIGHT STATE UNIVERSITY**

**November, 2021**

**REMEDIATION PLAN**

Recommendations in this report are based on the available findings from the phase 3 scanning results. In this process I am going to employ a risk-based, context-aware prioritization method. With this method I am going to focus on vulnerabilities that significantly posses higher risk than others. This is a tradeoff among a variety of considerations such as severity, fixability, coverage, and compliance. The plan recommends cost-effective safeguards to mitigate threats and associated exploitable vulnerabilities.

To that end, I would use the same severity ratings I employed in phase 3 to prioritize each remediation action.

**NAME: Weak Password Requirement**

**SEVERITY:** HIGH

**CVSS BASE SCORE: 8.1**

**CWE ID:** 521

**WEAKNESS ID:** 521

**OWASP 2017 CATEGORY:** A7-Broken Authentication

**AFFECTED HOSTS**: <http://10.0.2.4> (Login Authentication)

**VULNERABILITY DETAILS:** The web application does not have a password complexity requirement for signup functionality. This is a very high vulnerability that attackers can and like to exploit. It was also found that users are allowed to have the same password as their username, which is not in line with good security practice.

**ATTACK METHOD:** To test the password complexity policy of the application, I created a new login account with **username=” test email” and password =” admin”** to see if the application would suggest the creation of a stronger password to enforce a proper complexity policy.

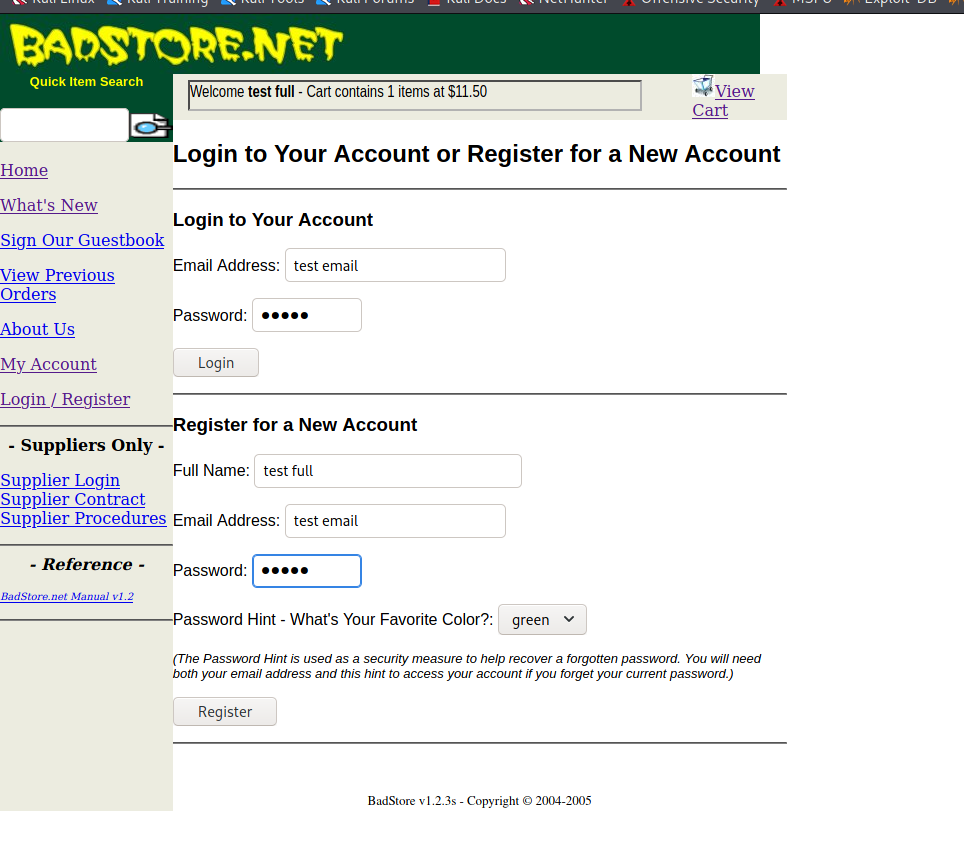
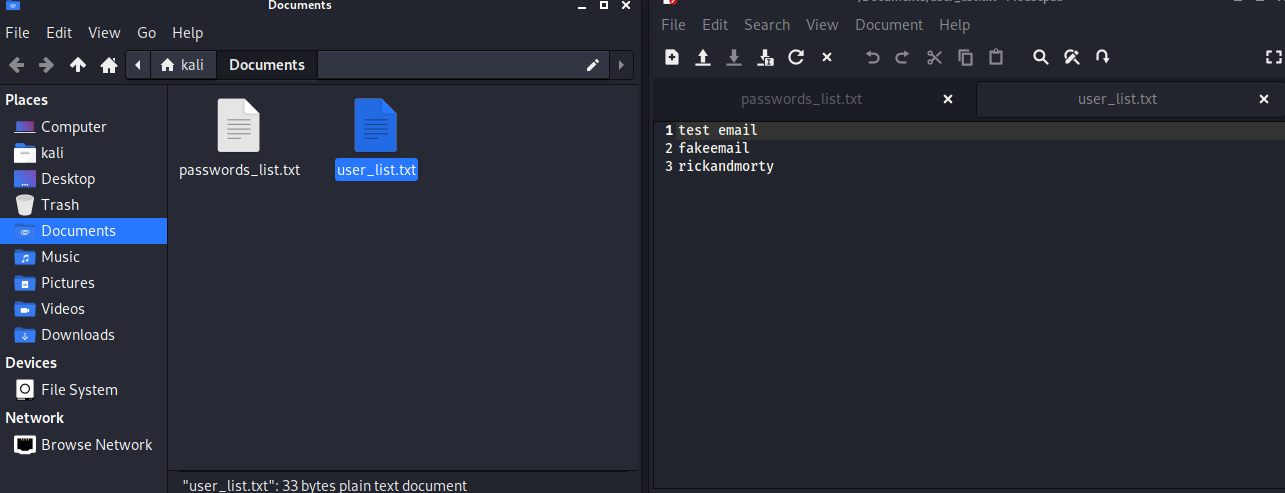
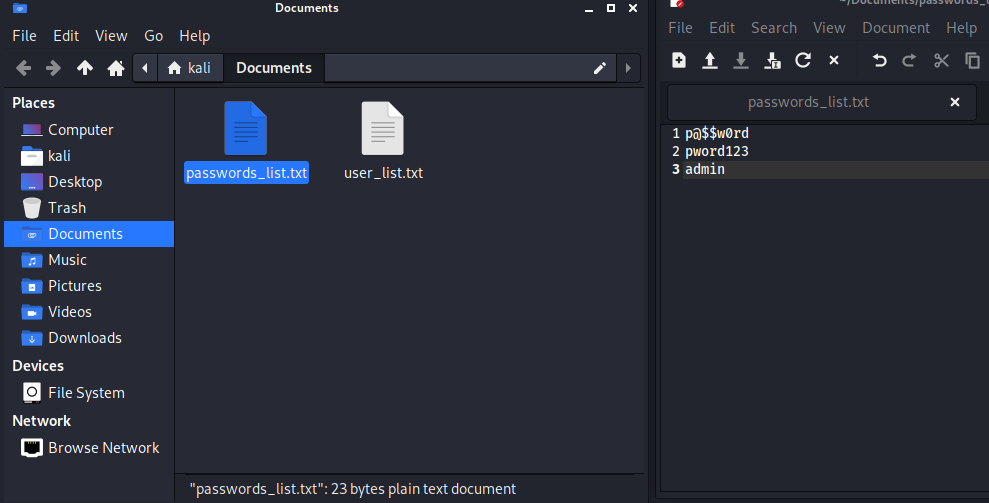


Figure 1. Application did not suggest a stronger password. Hence, generation of a weak password.

To initiate the attack, I used a password cracking application on Kali Linux called **Medusa** to use the brute force method. In the brute force method of password attacking, the concept is to try every possible combination of characters until a password is found.

I created 2 files (user\_list.txt and passwords\_list.txt) to store possible usernames and passwords, respectively.

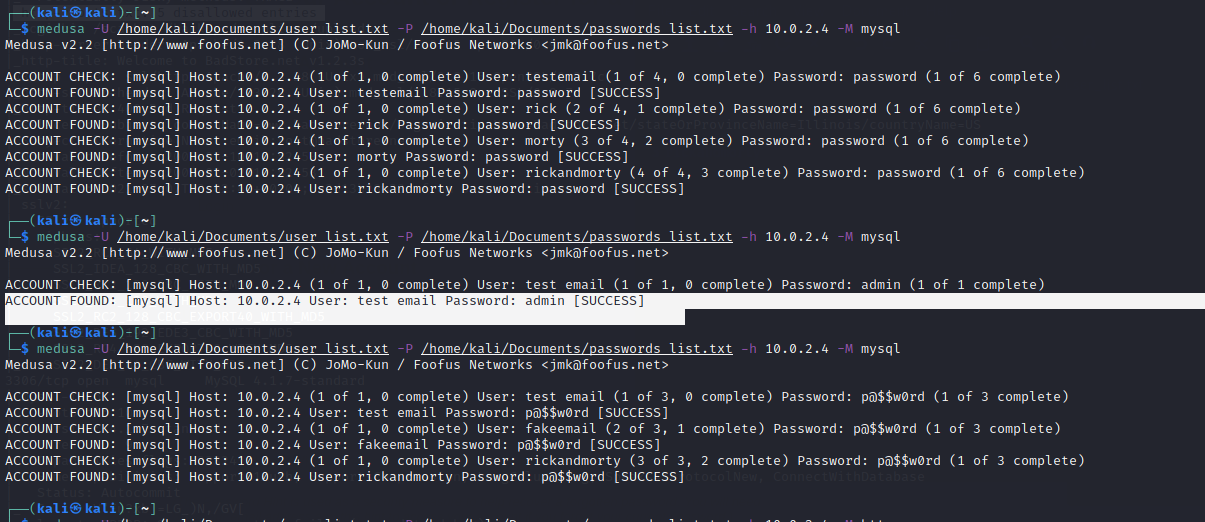
Figure 2.Possible Username list

Figure 3.Possible password list

Next, I ran this command **medusa -U /home/kali/Documents/user\_list.txt -P /home/kali/Documents/passwords\_list.txt -h 10.0.2.4 -M mysq**l

–U (location of username file) –P (location of password file) –h (the hosts IP address) –M (open service on host port to exploit)

After running the command, I got a success showing the username and the password of the target machine as shown below

Figure 4. Successful exploitation of vulnerability to get target username and password.

Successful exploitation of vulnerability to get target username and password through brute forcing.

**IMPACT:** Attackers may have access to unauthorized user information which may lead to money laundering, social security fraud, and identity theft, or disclose legally protected highly sensitive information.

**REMMEDIATION:** Align password length, complexity and rotation policies with NIST 800-63 B’s guidelines in section 5.1.1 for Memorized Secrets or other modern, evidence-based password policies. Also, limit failed login attempts. Implement multi-factor authentication.

**COST TO IMPLEMENT (Hr.):** 1

**RISK REDUCTION:** 10.0

**REMAINING RISK:** 0.0

**SECURITY CONTROL:** Implement a secure password policy that follows:

1. Minimum length requirement of 8 characters.

2. Upper- and lower-case characters must be used.

3. Minimum of 1 digit should be present in the password.

**NAME: PHPMYADMIN REFLECTED CROSS-SITE SCRIPTING**

**SEVERITY: HIGH**

**CVE: 2013-1937**

**CWE: 79**

**CVSS BASE SCORE: 6.1**

**AFFECTED HOSTS:** [**http://10.0.2.4**](http://10.0.2.4)

**VULNERABILITY DETAILS:** Reflected cross-site scripting (XSS) vulnerabilities are present on the host: <http://10.0.2.4> phpMyAdmin 3.5.x, before version 3.5.8. These allow remote attackers to inject arbitrary JavaScript or HTML via the visualizationSettings[width] or visualizationSettings[height] parameters.

**IMPACT:** Modification of some system files or information is possible, but the attacker does not have control over what can be modified, or the scope of what the attacker can affect is limited.

**REMEDIATION:** Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered, or escaped before echoing anything back to the user, such as the values of query parameters during searches.

**COST TO IMPLEMENT (Hr.):** 1

**RISK REDUCTION**: 8.0

**REMAINING RISK:** 2.0

**SECURITY CONTROL:** Upgrade phpMyAdmin to version 8.0.8 (latest as of 01 July 2021)

**NAME: Cross Site Scripting (DOM Based)**

**SEVERITY:** High

**CWE ID: 79**

**CVSS BASE SCORE: 7.6**

**OWASP 2017 CATEGORY:** A7-Cross Site Scripting (XSS)

**AFFECTED HOSTS**: <http://10.0.2.4> (Entire Site)

**VULNERABILITY DETAILS:** DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as JavaScript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible.

**IMPACT:** According to OWASP, the impact for DOM based XSS is moderate. It performs remote code execution on victim’s browser, such as stealing credentials, sessions, or malware delivery which is a disastrous consequence.

**REMMEDIATION:** Using a vetted library or framework such as Microsoft's Anti-XSS library, the OWASP ESAPI that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Graphical user interface, text, application, email

Description automatically generatedFigure 2. DOM Based XSS

**COST TO IMPLEMENT (Hr.):** 1

**RISK REDUCTION**: 7.0

**REMAINING RISK:** 3.0

**SECURITY CONTROL:** Ensure JavaScript variables are quoted, JavaScript Hex Encoding, JavaScript Unicode Encoding, avoid backslash encoding (\" or \' or \\).

**NAME: SQL Injection**

**SEVERITY:** High

**CWE ID:** 89

**CVSS BASE SCORE: 7.5**

**OWASP 2017 CATEGORY:** A1- Injection

**AFFECTED HOSTS**: http://10.0.2.4/cgi-bin/badstore.cgi?action=cartadd

**VULNERABILITY DETAILS:** The page results were successfully manipulated using the Boolean conditions [1000' AND '1'='1' -- ] and [1000' AND '1'='2' -- ]The parameter value being modified was NOT stripped from the HTML output for the purposes of the comparison. Data was returned for the original parameter. The vulnerability was detected by successfully restricting the data originally returned, by manipulating the parameter.

Graphical user interface, text, application

Description automatically generated

**IMPACT:** Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover. The business impact depends on the needs of the application and data.

**REMMEDIATION:** The preferred option is to use a safe API, which avoids the use of the interpreter entirely or provides a parameterized interface or migrate to use Object Relational Mapping Tools (ORMs). *Graphical user interface, text, application, email

Description automatically generatedFigure 5. SQL Injection*

**COST TO IMPLEMENT (Hr.):** 2

**RISK REDUCTION:** 7.7

**REMAINING RISK:** 2.3

**SECURITY CONTROL**: Minimizing least privileges assigned to every database account in the web application. Do not assign DBA or admin type access rights to application accounts.

**NAME: Application Error Disclosure**

**SEVERITY:** Medium

**CVSS BASE SCORE: 6.5**

**CWE ID:** 200

**OWASP 2017 CATEGORY:** A3-Sensitive Data Exposure

**AFFECTED HOSTS**: http://10.0.2.4/scanbot/

**VULNERABILITY DETAILS:** This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application.

Text

Description automatically generatedFigure 6. Evidence= "Parent Directory"

**IMPACT:** This information can be used to launch further attacks against the web application as well as steal important information.

**REMMEDIATION RECOMMENDATION:** Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.

**Text

Description automatically generated**

**COST TO IMPLEMENT (Hr.):** Total cost time is determined by the vendor.

**SECURITY CONTROL**: Policy for handling error should be documented for implementing a mechanism to provide a unique error reference/identifier to the client browser while logging the details on the server side and not exposing them to the user.

**NAME: Directory Browsing**

**SEVERITY:** Medium, Information

**CVSS BASE SCORE: 4.0**

**CWE ID:** 548

**OWASP 2017 CATEGORY:** A3-Sensitive Data Exposure

**AFFECTED HOSTS**: http://10.0.2.4/backup/

**VULNERABILITY DETAILS**: It is possible to view the directory listing. Directory listing may reveal hidden scripts, include files, backup source files, etc. which can be accessed to read sensitive information.

Text

Description automatically generated

**IMPACT**: Exposing the contents of a directory can lead to an attacker gaining access to source code or providing useful information for the attacker to devise exploits, such as creation times of files or any information that may be encoded in file names. The directory listing may also compromise private or confidential data.

**REMMEDIATION RECOMMENDATION:** Restricting access to important directories or files by adopting a need-to-know requirement for both the document and server root.

Graphical user interface

Description automatically generated with low confidence

**COST TO IMPLEMENT (Hr.):** 1.5

**SECURITY CONTROL**: Configure your web server to prevent directory listings for all paths beneath the web root; Place into each directory a default file (such as index.htm) that the web server will display instead of returning a directory listing.

**NAME: X-Frame-Options Header Not Set**

**SEVERITY:** Medium

**CWE ID:** 16

**AFFECTED HOSTS**: http://10.0.2.4

**VULNERABILITY DETAILS**: X-Frame-Options header is not included in the HTTP response to protect against 'Clickjacking' attacks.

Graphical user interface, text, application

Description automatically generated

**IMPACT:** inappropriate software configuration.

**REMMEDIATION RECOMMENDATION:** Ensure appropriate configuration of software.

Graphical user interface, text, application, email

Description automatically generated

**NAME: Absence of Anti-CSRF Tokens**

**SEVERITY:** Low

**CWE ID:** 352

**OWASP 2017 CATEGORY:** A8- Cross Site Request Forgery

**AFFECTED HOSTS**: http://10.0.2.4

**VULNERABILITY DETAILS:** No Anti-CSRF tokens were found in a HTML submission form. This forces a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim.Text

Description automatically generated with medium confidenceFigure 7. Evidence <FORM name=search onsubmit=/cgi-bin/badstore.cgi

**IMPACT:** Gain Privileges or Assume Identity; Bypass Protection Mechanism; Read Application Data; Modify Application Data; DoS attack.

**REMMEDIATION RECOMMENDATION:** Use anti-CSRF packages such as the OWASP CSRF Guard

Graphical user interface, application

Description automatically generated

**NAME: Cookie No HttpOnly Flag**

**SEVERITY:** Low

**CWE ID:** 16

**OWASP 2017 CATEGORY:**

**AFFECTED HOSTS**: <http://10.0.2.4> (Entire Site)

**VULNERABILITY DETAILS:** A cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript. If a malicious script can be run on this page, then the cookie will be accessible and can be transmitted to another site. If this is a session cookie then session hijacking may be possible.

Graphical user interface, text, application, email

Description automatically generated

Figure 8.Evidence. Set-Cookie

**IMPACT:** Session hijacking may have disastrous consequences on user account.

Graphical user interface, text, application

Description automatically generated

**NAME: Private IP Disclosure**

**SEVERITY:** Low

**CWE ID:** 200

**OWASP 2017 CATEGORY:** A3-Sensitive Data Exposure

**AFFECTED HOSTS**: http://10.0.2.4/cgibin/badstore.cgi?action=cartadd&action=search&searchquery=ZAP

**VULNERABILITY DETAILS:** A private IP (10.0.2.15) has been found in the HTTP response body. This information might be helpful for further attacks targeting internal systems.

Text

Description automatically generated with medium confidenceFigure 9. Exposed IP address is a potential threat

**IMPACT:** Severe damage to the confidentiality of information. Data can be read and accessible to unauthorized parties.

**REMMEDIATION RECOMMENDATION:** Compartmentalize the system to have "safe" areas where trust boundaries can be unambiguously drawn. Do not allow sensitive data to go outside of the trust boundary and always be careful when interfacing with a compartment outside of the safe area.

Graphical user interface, text

Description automatically generated