**Guardian Store Vulnerability Report**

**Table of Contents**

[**Abstract** 3](#_Toc147591487)

[**Overview** 4](#_Toc147591488)

[**Introduction** 4](#_Toc147591489)

[**Vulnerabilities Identified** 4](#_Toc147591490)

[**Chosen Vulnerability:** Broken Authentication 5](#_Toc147591491)

[**Introduction** 5](#_Toc147591492)

[**Research** 5](#_Toc147591493)

[**Exploitation** 5](#_Toc147591494)

[**Patch** 5](#_Toc147591495)

[**Security Opportunities** 6](#_Toc147591496)

[**Defence-In-Depth** 6](#_Toc147591497)

[Definition 6](#_Toc147591498)

[Opportunities 6](#_Toc147591499)

[**Secure Software Development Mitigations** 6](#_Toc147591500)

[Definition 6](#_Toc147591501)

[Opportunities 6](#_Toc147591502)

[**Conclusion** 7](#_Toc147591503)

[**Bibliography & References** 8](#_Toc147591504)

# **Executive Summary**

# **Overview**

## **Introduction**

Guardian has contacted us to help penetration test GuardianStore, identify vulnerabilities and attempt to resolve them.

## **Vulnerabilities Identified**

The following vulnerabilities have been identified:

* Broken Authentication/Broken Access Controls
* Broken authorization
* Business logic flaws
* Cross-site request forgery
* Cross-site scripting
* Cryptography issues
* Insecure file upload
* NoSQL injection
* Sensitive data exposure
* SQL injection

# **Chosen Vulnerability:** Broken Authorization/Broken Access Control: Password Strength

## **Introduction**

The chosen vulnerability class for this penetration report is Broken Authorization, also known as Broken Access Control. This is a major vulnerability class which could compromise the account security and confidentiality of clients and admins alike. The chosen vaulerability is password strength and current weakness to brute force attacks. Being a surprisingly common vulnerability, finding a fix for this issue will mean similar cases and issues can also be resolved.

## 

## **Research**

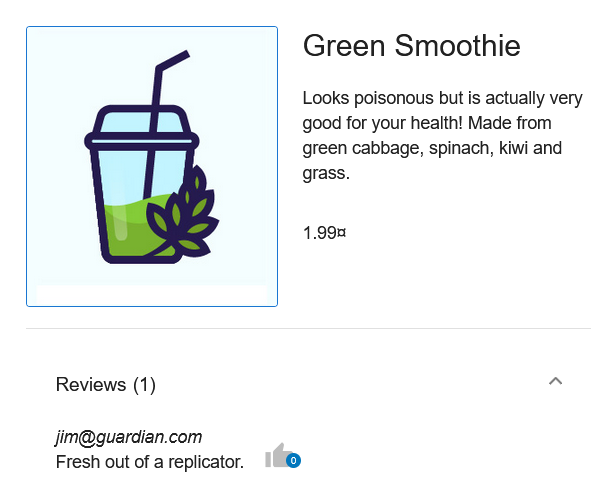
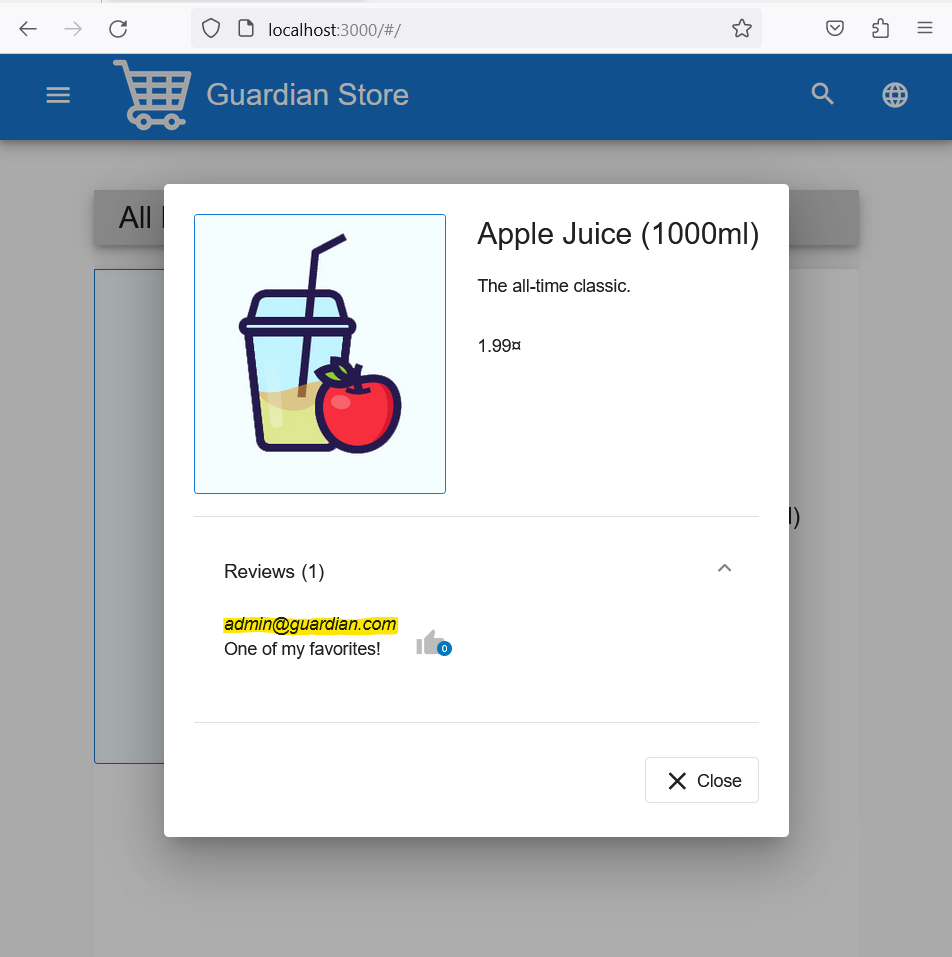
Broken

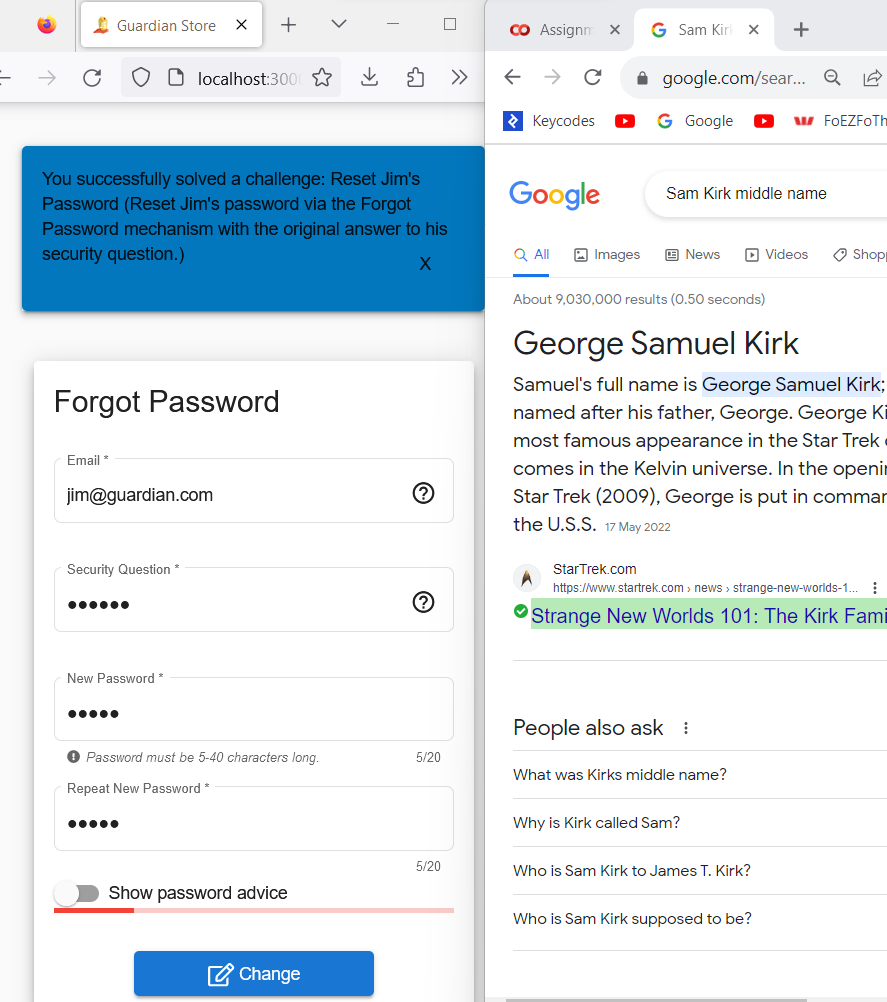
## **Reconnaissance**

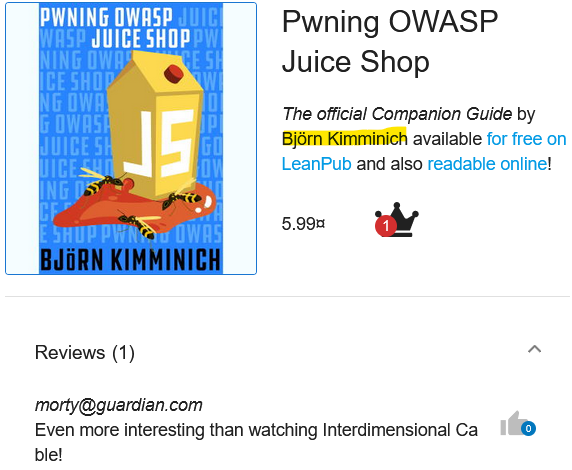
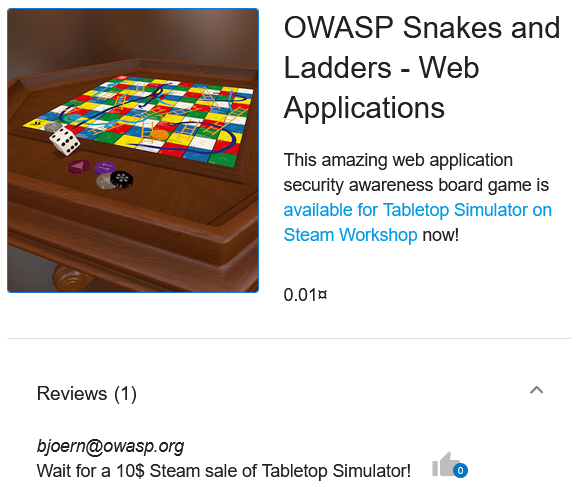
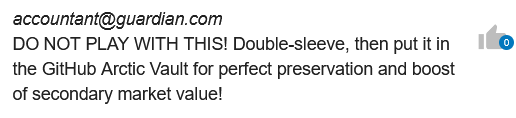
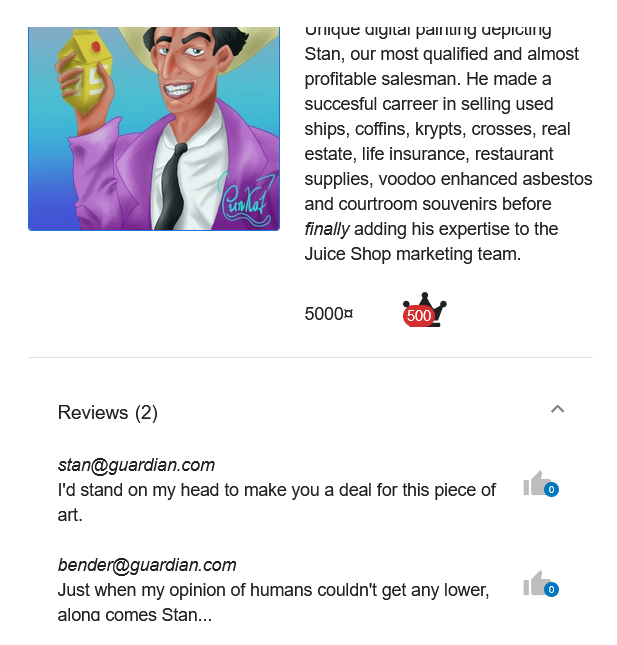
A screenshot of a login form

Description automatically generatedIn order to break the authorization and login, we require an email and password, as an attacker an admin email or an email with high privileges is preferred. However gaining access to as many accounts as possible is also beneficial because we can exploit broken authorization to elevate unprivileged accounts privilege.

To find accounts email addresses, I began combing through reviews of products on the store page. After reviewing them all, I found an admin account with the email address [admin@guardian.com](mailto:admin@guardian.com). As well as [jim@guardian.com](mailto:jim@guardian.com), [bender@guardian.com](mailto:bender@guardian.com), [uvogin@guardian.com](mailto:uvogin@guardian.com), [stan@guardian.com](mailto:stan@guardian.com), [bjoern@owasp.org](mailto:bjoern@owasp.org), [morty@guardian.com](mailto:morty@guardian.com), [mc\_safesearch@guardian.com](mailto:mc_safesearch@guardian.com), and [accountant@guardian.com](mailto:accountant@guardian.com) user accounts.



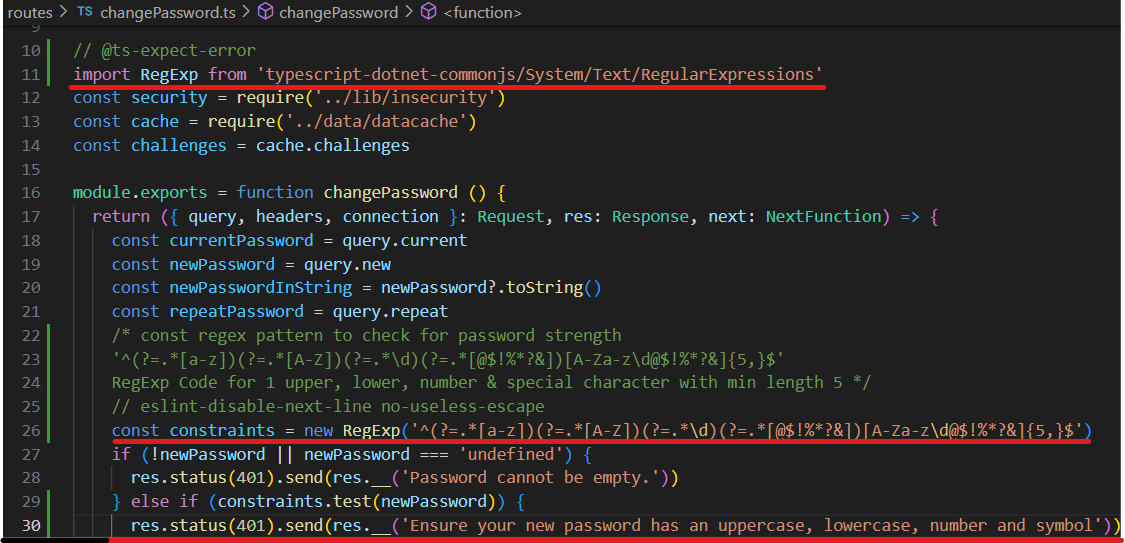
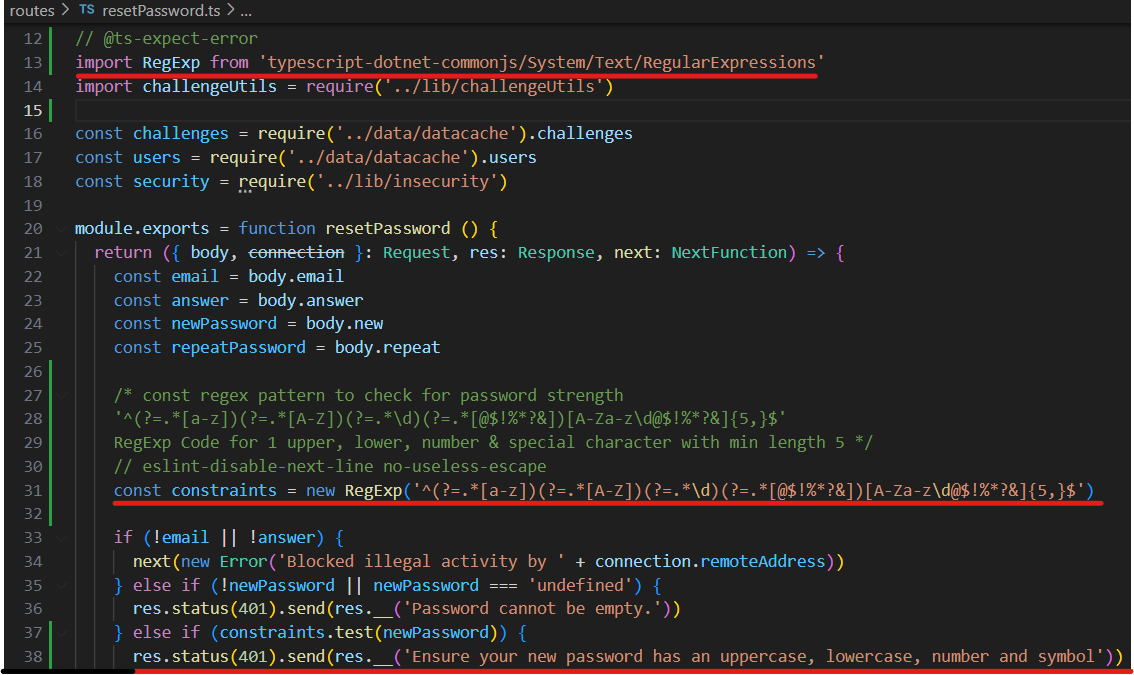
This made it easy to find accounts and left me with just having to get or reset the password. This could be gotten by using social engineering, or brute force attacks, to guess the password. As well as, resetting the password, by answering the security question.  
This reconnaissance vulnerability could be patched with the username showing instead of the address i.e., “Jim” instead of [jim@guardian.com](mailto:jim@guardian.com).



## **Exploitation**

To gain access to the admin account I first tried SQL Injection of “ ’ ” and “ ‘ OR 1=1-- ” as the password with little luck, other than prompting an error handling exception. This led me to doing a most common admin passwords brute force sniper payload attack on the admin account, as well as a most common surnames brute force sniper payload attack. This is because having access to both the password, or the means to change the password, would give me access to the account. The more boarder the attack, the more avenues used, and more surfaces exploited, the more likely for a successful breach. I hand designed the password payload to include common weak 5 digit admin passwords like; roots, admin, 12345, etc with variations including capital letters and number strings. Eventuality the password Admin123 was found to be correct, grating me access to the admins account and allowing me to change the password.

## **Patch**

To mitigate against the password strength authentication vulnerability, we should implement a password strength test before allowing the resetting or changing of a password. This can be done rather simply using RegExp which allows you to create a character list to check upon a string. By using a list which ensures the password is atleast 5 characters long with a lowercase, uppercase, number and symbol.  
This was added to the reset & change password function, ensuring that old insecure passwords are phased out with new secure ones in time. While simultaneously ensuring uses don’t use weaker passwords like Ca$h1, as they are wont to do, when forced to change their password.

Furthermore, by adding a notification informing users to update their passwords, manually updating the admins passwords and by utilising defence-in-depth to enforcing a strict internet security policy among our staff. we can ensure weak admin passwords like Admin123 don’t exist. In effect patching out the password strength vulnerability.

In the future we could further protect against brute force attacks, by implementing a max access attempt threshold, which when passed, would lock down the account. Stopping a brute force payload attack dead in its tracks.  
We would also need to implement an unlock mechanism, to ensure users can contact support, have their password changed and their account unlocked.   
Multi-Factor Authentication through an Email or SMS authentication request, requiring validation, would mitigate the attack. As third-parties would be required to have access to the request source, in order to get access to the account.

# **Security Opportunities**

System & Application

Network

Physical

## **Defence-In-Depth**

### Definition

Defence in depth is a multi-layered security control methodology devised by the National Security Agency (NSA), and is a globally prevalent standard, for cyber security and data protection. It’s aim is to supply redundancy, procedural, technical, personal, and physical security.

Defence-In-Depth is often segmented into physical, technical, and administrative controls.  
**Physical:** physical controls, defending a system i.e., guards, locks, fences, security, etc.  
**Technical:** technical controls, securing a system i.e., Firewalls, Authentication, etc.  
**Administrative:** administrative controls, protecting a system i.e., policies, procedures, etc.

The layers that make up Defence-In-depth are System & Application, Network, and Physical. This includes but isn’t limited to the following methods:

|  |  |  |
| --- | --- | --- |
| **System and application:**  Antivirus software Multi-factor authentication  Encryption Hashing passwords Vulnerability scanners Timed access control Internet Security Awareness | **Network:**  Firewalls (hardware or software) Demilitarized zones (DMZ) Virtual private network (VPN) | **Physical:**  Biometrics Data-centric security Physical security |

### Opportunities

**Technical:** System, Application and NetworkImplementing Multi-factor Authentication would be a great opportunity to fix our broken authentication & authorisation.Encryption and password hashing should also be look into to secure our backend data and mitigate man-in-the-middle attack scouting.

**Administrative:** System, Application and Physical

An Administrative Defence-In-Depth opportunity would be teaching our administrators Internet Security Awareness. By learning common tactics, threats and red flags for social engineering or other attack methods, we can mitigate their effectiveness at tricking our staff into compromising system security. Intern further strengthening our system.

**Physical:** Network and Physical

To Physically protect our system we would implement firewalls between our servers on the Network layer and on the Physical layer we should implement site security protocols with guards and surveillance to ensure no physical harm or improper access can happen on our system

## **Secure Software Development Mitigations**

### Definition

Secure Software Development Principles which make recommendations on how to strengthen our cyber security and mitigate against attacks or intrusions..

**Principles:**

* Least Privilege

Users have the minimum privilege required for proper function.

* Fail-Safe Defaults

Defaults should be created that maintain security, in case of failure.

* Economy of Mechanism

Simple security design and controls, the simpler the more stable.

* Complete Mediation

Access must be authorised before granting access.

* Open Design

Security can’t depend on secrecy of design or implementation.

* Separation of Privilege

Security checks should require multiple conditions to be met. More secure than one condition.

* Least Common Mechanism

Minimise commonalities and dependencies between users.

* Psychological Acceptability

Security controls should be easy-to-use, understand, and control, rather than bypass.

### Opportunities

# **Conclusion**

By implementing \_\_ we can fix the broken authentication. Though making use of opportunities presented to us by Defence-In-Depth methods and Secure Software Development Mitigation Techniques we can further defend our system from future attacks.

# **Bibliography & References**

[1] OWASP. (2021). Secure Product Design - OWASP Cheat Sheet Series. [online] Available at: [https://cheatsheetseries.owasp.org/cheatsheets/Secure\_Product\_Design\_Cheat\_Sheet.  
html#2-the-principle-of-defense-in-depth](https://cheatsheetseries.owasp.org/cheatsheets/Secure_Product_Design_Cheat_Sheet.html#2-the-principle-of-defense-in-depth).

[2] NSA .(2012). Defence In Depth. [online] Available at: [https://web.archive.org/web/  
20121002051613/https://www.nsa.gov/ia/\_files/support/defenseindepth.pdf.](https://web.archive.org/web/20121002051613/https://www.nsa.gov/ia/_files/support/defenseindepth.pdf.)

[3] A., Barrett. (2015). Some principles of secure design – Designing Secure System module Autumn 2015. [online] Available at: <https://docplayer.net/17241198-Some-principles-of-secure-design-designing-secure-systems-module-autumn-2015.html>

[4] OWASP. (2021). A01 Broken Access Control - OWASP Top 10:2021. [online] owasp.org. Available at: <https://owasp.org/Top10/A01_2021-Broken_Access_Control/>.

[5] Miessler, D. (2022). Daniel Miessler SecLists Top 10000 Passwords. [online] GitHub. Available at: <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-10000.txt>.

[6] PowerShell Gallery (2023). PowerShell Gallery | Surnames.txt 0.1.1. [online] Available at: <https://www.powershellgallery.com/packages/TelligentCommunitySample/0.1.1/Content/Surnames.txt>.

[7] OWASP (2021). OWASP Top Ten. [online] Owasp.org. Available at: <https://owasp.org/www-project-top-ten/>.