**Hackathon Juice Shop**

*By Oluwamayowa Akinola, Alvis Auzins & Samuel Dauda*

A01:2021 - Broken Access Control

Problem:

When users have permissions higher than what they are intended to have. This could be including but not limited to admin capabilities without being admin.

Problems can also occur due to misconfiguration and programming error.

Another factor is when the security measures are not properly implemented.

[(OWASP, 2021a)](https://www.zotero.org/google-docs/?ICJekV)

Forces:

There are multiple reasons where this issue could have arisen from. Some examples: A user is assigned a role that they are not intended to have - A low level user being assigned the role of admin for example.

Another possibility could be users are able to edit the url to allow them to bypass security checks.

Both of these as well as other not mentioned have the potential to allow users to see and also manage aspects that they are not supposed to be able to do

[(OWASP, 2021a)](https://www.zotero.org/google-docs/?8uJqSJ)

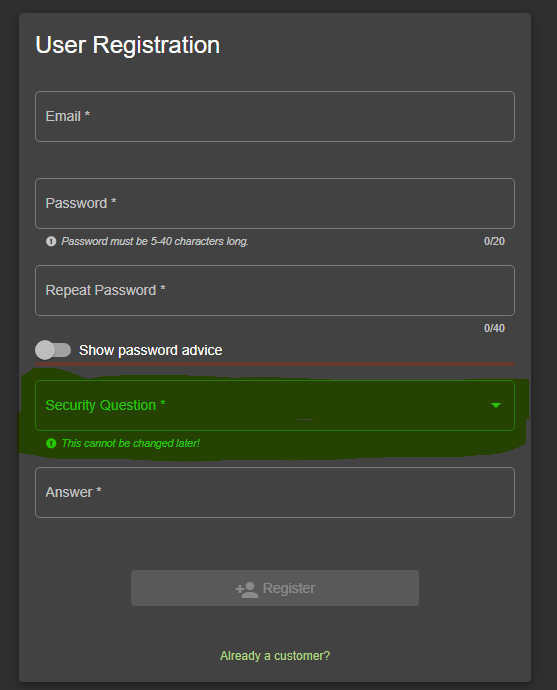
In this project this aspect is showcased in the login section of the application. There are 2 notable aspects here that makes this application vulnerable to this type of security issue.

1. This application does not enforce a strong password when users are creating their account. The application suggests that users create a strong password by using a mixture of capital letters, lowercase letters,numbers, special characters as well as having a password that's 8 digits long. Unfortunately the application does not enforce this and users are able to create a password of a minimum 5 digits. All of which with the same character if wanted. Users are able to create accounts with extremely weak passwords which leaves the account vulnerable to wrong users accessing it.

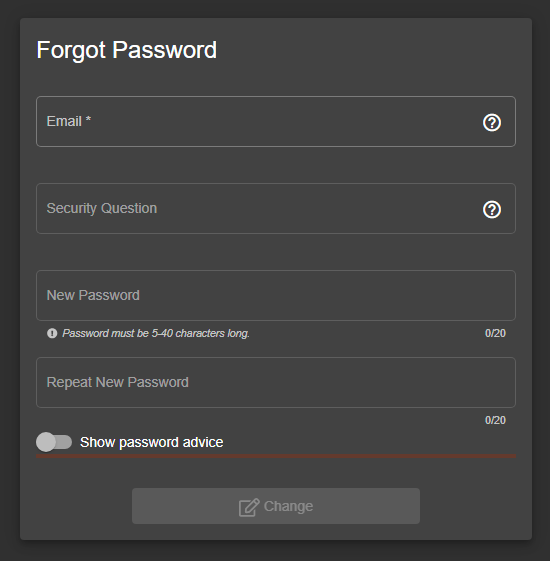
(The password inserted here is “aaaaa”. As it can be seen the application is accepting this as a suitable password)


# (The password inserted here is “aaaaa”. As it can be seen the application is accepting this as a suitable password)

1. Another example that is showcased here is the use of security questions. The use of security questions is not an ideal method of authentication as it is not secure and information requested can sometimes be easily located by the wrong user. A malicious user would be able to easily access the account if they know the email address of the account by utilising the forget password option.



# (Creating account allows the user to add a security question to the account)



# (It showcased here that users are able to change the password of the account by just knowing the security question)

Solution:

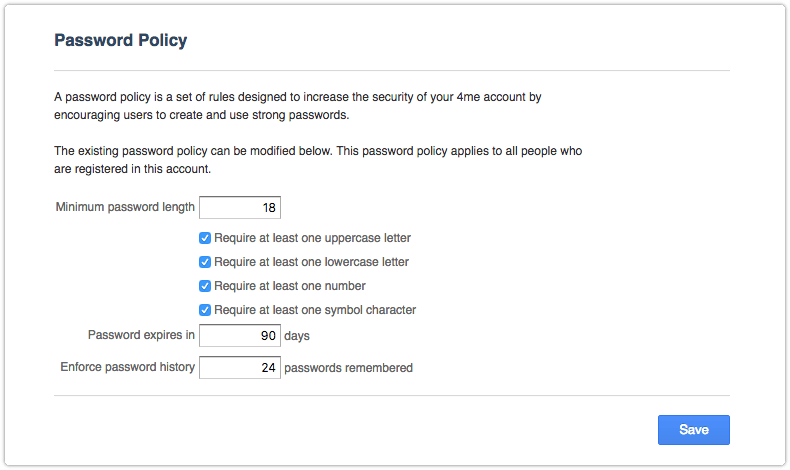
Several solutions can be employed, the principle of least privilege can be implemented, which ensures that each user has only the minimum level of access required to perform their job.

Strong authentication policy can also be implemented to enhance access control such as multi-factor authentication mechanisms. The example of multi-factor authentication can be something the user has (smartcard), something the user is (biometric) , something the user knows (password) and something the user does ( signature).

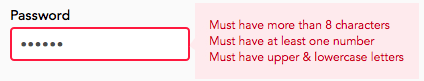
Regular security audits and monitoring can help identify and rectify access control vulnerabilities.]

In this project there are multiple options we can use to tackle the showcased problems

1. It is suggested that password strength should be enforced. Weak passwords can make applications vulnerable. We can see that the application is already aware of what a strong password would entail (capital letters, lowercase letters,numbers, special characters, 8 digits long). By making the application require that all users' passwords follow this advice it would make the application more secure.

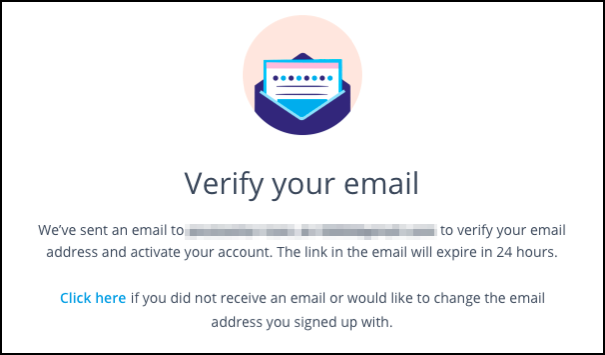
[](https://www.4me.com/blog/password-policy/)

# (This is an example of how an application can enforce a password policy to its users [(Prins, 2018)](https://www.zotero.org/google-docs/?sQTHno))

[](https://passwordpolicies.cs.princeton.edu/)

# (This is an example of users trying to create a password that doesn't follow the policy [(Lee et al., 2022)](https://www.zotero.org/google-docs/?F3Jpug))

1. Instead of using security questions which are known to be vulnerable, the application should use Two Factor Authentication(2FA) to verify users and strengthen its security. There are many different types of 2FA that an application can use - Email verification, mobile number verification and application verification such as google authenticator app to name a few [(Microsoft Security, n.d.)](https://www.zotero.org/google-docs/?GoNj8n). The easiest method of these to implement would be email verification. The application already requires the user to utilise an email address when creating the account. The application could require the user to input a randomly generated code, which would be sent to the email address when creating the account to confirm the user. By doing this the application is not only confirming that the user has access to the email address that is being utilised but is also able to confirm users identity by using this same method again in the future, for example using this method when a user forgets their password

[](https://help.adroll.com/hc/en-us/articles/360057993352-Email-Verification-Verifying-your-email-address-when-you-sign-up)

# (Example image of a website requesting user to verify their email address [(AdRoll, 2024)](https://www.zotero.org/google-docs/?9HcW0P))

Consequences:

When access control mechanisms fail, unauthorised users can gain access to sensitive information, leading to data breaches, financial losses, and damage to the organisation’s reputation.

Broken access control can result in the manipulation or destruction of the critical dada, impacting business operations and disrupting services.

It can also lead to compliance violations, legal repercussions, and regulatory fines.

Related Aspects:

Security requirements and recommendations is a topic that is constantly evolving. As new solutions and techniques appear, new vulnerabilities and exploits also appear. To keep the application secure, the client would need to keep up-to-date with password/user login recommendations, whether these recommendations are 2FA, the use of keys or future techniques that are not yet present.

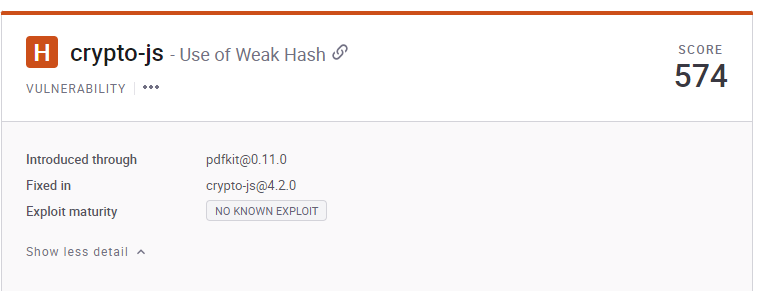
A02:2021 - Cryptographic Failures

Problem:

1. Weak algorithms: The use of outdated or weak cryptographic algorithms can make it easier for attackers to break the encryption.
2. Poor key management: Inadequate key management practice such as using weak keys, reusing keys, or strong keys insecurely can lead to cryptographic failures.
3. Implementation flaws: Errors in the implementation of cryptographic protocols or libraries can create vulnerabilities that attackers can exploit.

Forces:

Cryptographic failures pose significant risks to data security and privacy. Various actors, including hackers, malicious insiders, and state-sponsored entities, can exploit vulnerabilities in cryptographic systems. Technical constraints such as weak encryption algorithms, flawed implementations and insufficient key management can also contribute to cryptographic failures. Additionally, technology constraints such as hardware limitations and compatibility issues can impede the effectiveness of cryptographic solutions.



# (Screenshot from Snyk showcasing a vulnerability)

Solutions:

1. Adoption of Strong Cryptographic Algorithms: Ensure that the cryptographic algorithms used in the system are strong and resistant to known attacks as well as regularly update cryptographic protocols and algorithms to stay ahead of emerging threats.

According to Snyk overview in the above screenshot:

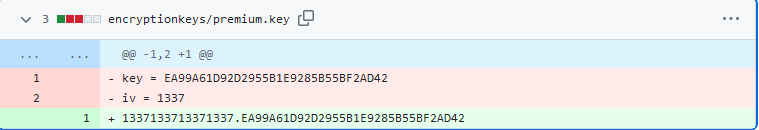
[crypto-js](https://github.com/brix/crypto-js) is a library of crypto standards [(Brix, 2024)](https://www.zotero.org/google-docs/?DGCOdb).

Affected versions of this package are vulnerable to Use of Weak Hash due to inadequate security settings in the PBKDF2 configuration, which uses insecure SHA1 and has a low iteration count of 1. These insecure settings allow attackers to perform brute-force attacks when PBKDF2 is used with the default parameters.

No information is directly exposed when a hash is generated, regardless of whether the PBKDF2 function is in the vulnerable configuration or not. However, it may be possible to recover the original data, more or less easily depending on the configured parameters, using a brute force attack. This has a low impact on the confidentiality of the protected data, which are in a different scope than the vulnerable package.

The attacker similarly may be able to modify some data which is meant to be protected by the vulnerable package - most commonly when it is used for signature verification. This would require a subsequent exploitation, such as forcing a hash collision via length extension attack. The integrity of the data is therefore compromised, but the quantity and targeting of that data is not fully in the attacker's control, yielding a low integrity impact. According to the crypto-js maintainer: "Active development of CryptoJS has been discontinued. This library is no longer maintained." It is recommended to use the Node.js native crypto module.

1. Secure Key Management: Implement proper key generation, storage, and distribution practices to prevent unauthorised access to keys. Utilise secure key storage mechanisms and regularly rotate keys to minimise the impact of potential key compromises.

****

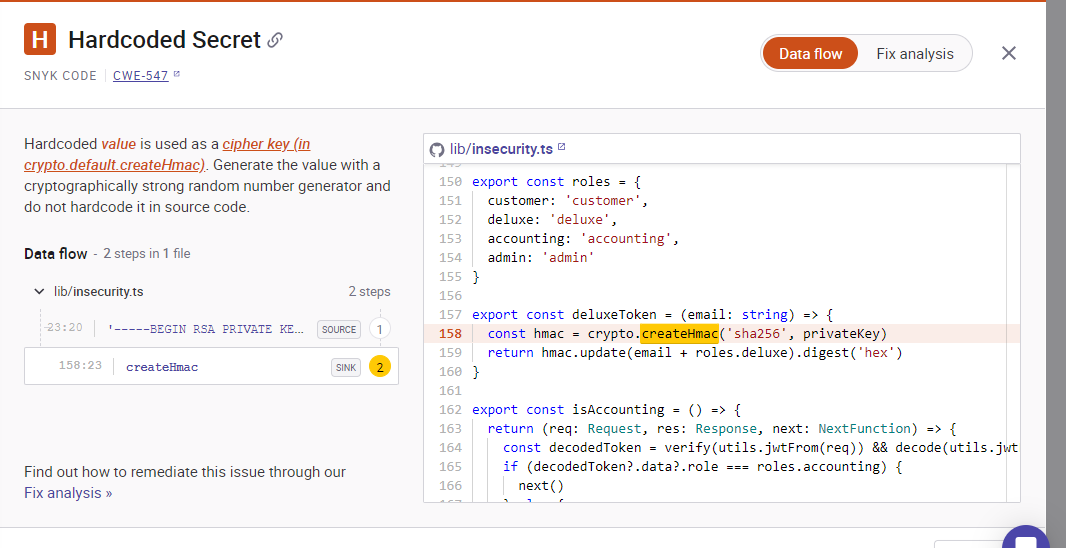
# (The above screenshot is taken from the Juice\_shop application, it identified the exposed encryption keys)

The above screenshot is from the juice-shop application, **Identified the exposed encryption keys,** it shows that in this application the encryption-keys are not being isolated from the application code and data. The keys should have been stored separately in a secured location, such as a key management system. Exposing the encryption keys makes the application vulnerable to attack. It should have been prevented from unauthorised users for potential exploit.



# (Screenshot from Snyk showcasing a vulnerability)

[jsonwebtoken](https://github.com/auth0/node-jsonwebtoken) is a JSON Web Token implementation (symmetric and asymmetric) [(auth0, 2024)](https://www.zotero.org/google-docs/?YHSoeJ). Affected versions of this package are vulnerable to Use of a Broken or Risky Cryptographic Algorithm such that the library can be misconfigured to use legacy, insecure key types for signature verification. For example, DSA keys could be used with the RS256 algorithm.

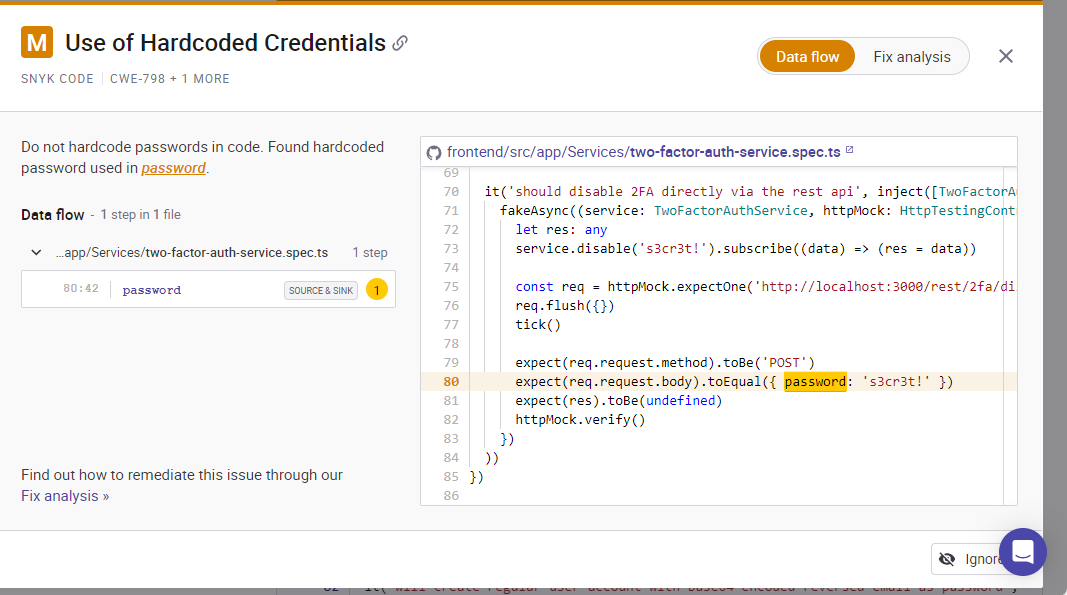


(Hardcoded Secret vulnerability detected by Snyk in the above screenshot)

The above screenshot shows another cryptographic failure. According to the scan done with the Snyk, it reveals that when constants are hardcoded into applications, this information could easily be reverse-engineered and become known to attackers. For example, if a breached authentication token is hardcoded in multiple places in the application, it may lead to components of the application remaining vulnerable if not all instances are changed. Another negative effect of hard-coding constants is potential unpredictability in the application's performance if the development team fails to update every single instance of the hardcoded constant throughout the code. For these reasons, hard-coding security-relevant constants is considered bad coding practice and should be remedied if present and avoided in future.

Best practices for prevention

* Never hard code security-related constants; use symbolic names or configuration lookup files.
* As hard coding is often done by coders working alone on a small scale, examine all legacy code components and test carefully when scaling.
* Adopt a "future-proof code" mindset: While use of constants may save a little time now and make development simpler in the short term, it could cost time and money adapting to scale or other unforeseen circumstances (such as new hardware) in the future.



# (The screenshot above is from Snyk, it revealed the vulnerability in the hardcoded credentials)

Details of the screenshot above:

The above scan with Snyk explained that developers may use hardcoded credentials for convenience when coding in order to simplify their workflow. While they are responsible for removing these before production, occasionally this task may fall through the cracks. This also becomes a maintenance challenge when credentials are re-used across multiple applications.

Once attackers gain access, they may take advantage of privilege level to remove or alter data, take down a site or app, or hold any of the above for ransom. The risk across multiple similar projects is even greater. If code containing the credentials is reused across multiple projects, they will all be compromised.

* Plan software architecture such that keys and passwords are always stored outside the code, wherever possible.
* Plan encryption into software architecture for all credential information and ensure proper handling of keys, credentials, and passwords.
* Prompt for a secure password on first login rather than hard-code a default password.
* If a hardcoded password or credential must be used, limit its use, for example, to system console users rather than via the network.
* Use strong hashes for inbound password authentication, ideally with randomly assigned salts to increase the difficulty level in case of brute-force attack.

1. Thorough Security Testing: Conduct comprehensive security testing and cryptographic assessment to identify vulnerabilities and weakness in the cryptographic components of the system.
2. Adherence to Best Practices: Follow industry best practices for cryptographic implementation, such as those outlined by standards organisations ;like NIST, IETF and others .

**Consequences:**

The implementation of these solutions to cryptographic failures will bring numerous benefits and improvements.

Firstly, it will enhance data security by mitigating vulnerabilities in encryption algorithms, safeguarding sensitive information from unauthorised access. Secondly, it will bolster trust in digital transactions and communication, fostering a more secure online environment for individuals and businesses. Moreover, addressing cryptographic failures will drive innovation in cybersecurity, leading to the development of more robust encryption techniques and protocols. Overall, this initiative will fortify the foundation of digital, security, integrity, and authenticity of data in an increasingly interconnected world.

Related Aspects:

Cryptographic failure can have significant implications for security. In addition to technical considerations, it is crucial to address human factors in system design and implementation to prevent vulnerabilities. Properly referencing established cryptographic standards and best practices is essential for ensuring the reliability of cryptographic solutions. As technology evolves, future changes such as quantum computing may require the adoption of new encryption methods to safeguard data effectively. Continuous monitoring and adaptation to emerging threats and advancements in cryptography research will be key to mitigating the risks associated with cryptographic failures and maintaining secure communication channels [(OWASP, 2021b)](https://www.zotero.org/google-docs/?kSP0Mg)**.**

A03:2021 - Injection

Problem:

When user-supplied data is entered and is not validated,filtered or sanitised by application. When these steps aren’t taken, it can cause the application to function in ways that it is not intended to function. For example it can cause the application to reveal sensitive information that shouldnt be showcased to the public or allow actions to be taken that shouldn't be allowed to happen, such as deleting a database [(OWASP, 2021c)](https://www.zotero.org/google-docs/?3U20b3).

Forces:

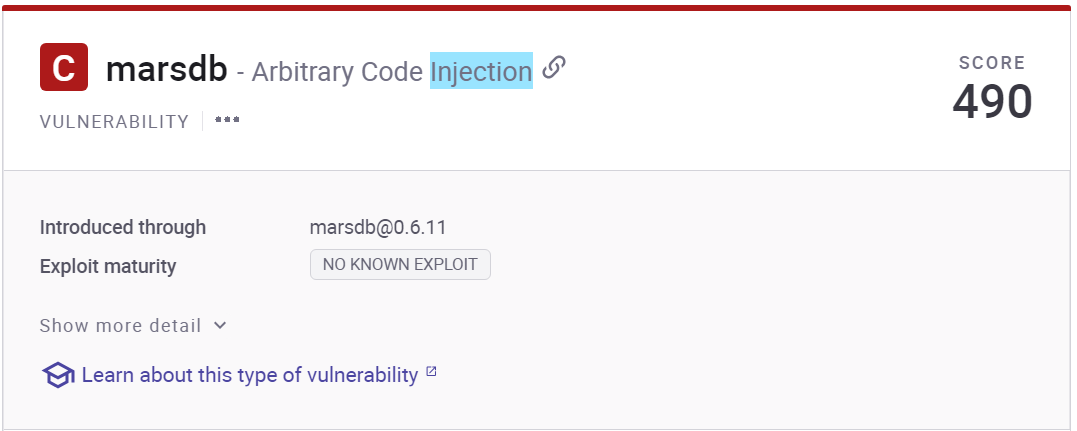
Not validating the application could put the application at risk as the user’s personal application could be modified to function in unique ways. Server side validation allows for safety as it will allow for modified changes to be noticed and restricted.[(OWASP, 2021c; Snyk Learn, 2024)](https://www.zotero.org/google-docs/?2PQUcJ)

Solutions:

Potential solutions: Using server side validation would allow the application to compare its functionality with that that is available on the server. This would help prevent injection techniques as it would be able to compare the functionality. Using LIMIT and other SQL commands to reduce the damage caused by the injections. Source code reviews and parameters should be automated tested (Such as HEADER, URL cookies, JSON)

We can see that currently, after using an automated process such a Snyk [(Snyk, 2024)](https://www.zotero.org/google-docs/?vJEtUv) to analyse the project, it is vulnerable to multiple types of injection techniques. It is vulnerable to command injection, arbitrary code injection.

# (Snyk screenshot showcasing application has a command injection vulnerability )



# (Screenshot from Snyk showcasing application has a arbitrary code injection vulnerability)

By using Snyk, it gives us potential solutions for these issues. For the command injection exploit, it mentions that a specific version of the package “sanitize-html” [(Apostrophe Technologies, 2024)](https://www.zotero.org/google-docs/?JheBvo) that is used in this application is vulnerable to this type of injection. So by upgrading the package to version recommend, it would resolve this vulnerability due to the fact that the newer version has rectified the issues

For the second example of the vulnerability showcased Snyk mentions the vulnerability through the “marsdb” package [(c58 and Artemyev, 2016)](https://www.zotero.org/google-docs/?6rwLou). After a brief analysis of the package, we can see that there is no newer version of the package, meaning that we can’t attempt to update it to check if the vulnerability is still there. There are 2 potential outcomes we can take to rectify this issue.

1st would be to remove the package and use a different package instead. By doing this we can choose a package that doesn't have this vulnerability but also a package that is still being updated and tested as the “marsdb” package that is being used here has not been updated in the last 8 years.

The 2nd option would be to analyse the source code of the files that use this package and see if the function that makes this package vulnerable is present. Snyk tells us that the use of the “*$where*” clause is what causes the vulnerability as it does not sanitise the code. While this solution is viable, it is recommended that a different package be used due to the fact that packages that aren’t still in its lifecycle could be exposed to more vulnerabilities down the line and lack of updates could leave the application vulnerable once again

Consequences:

Upgrading the package to a newer version will remove the command injection vulnerability, securing the application once more. By also using a different package instead of the “Marsdb” package [(c58 and Artemyev, 2016)](https://www.zotero.org/google-docs/?RCcjcg), we not only remove the arbitrary code injection vulnerability that may be also present but we further secure the application and future proof it by using a package this in its lifecycle and getting regular updates

Related Aspects:

Monitoring is important to allow the client to be notified if an injection exploit has been attempted or completed successfully. For example when a SQL injection is being attempted it is unlikely that the attacker is familiar with the names of the databases used. Due to this it is typical that errors are triggered. A SIEM can be used to automatically detect these errors and attempts of exploitation and rectify the issue before the attacker becomes successful [(IBM, 2024a; Microsoft Learn, 2023a)](https://www.zotero.org/google-docs/?Me9Su3). Quick reactions to these types of attacks are important as the exploit can be stopped before the attacker has caused any damage or has acquired important data.

[(IBM, 2021; SolarWinds, n.d.)](https://www.zotero.org/google-docs/?8TN9h4)

There are multiple methods that the client can use to prevent future vulnerabilities of injection. A developer security platform such as Snyk [(Snyk, 2024)](https://www.zotero.org/google-docs/?natYrP) can be used. It will scan the application for vulnerabilities which would allow the client to proactively prevent and rectify vulnerabilities. A security linter could also be used to help prevent this. This software analyses source code for vulnerabilities and potential error messages. Using both of these will further strengthen the application. [(OWASP, 2021c; Snyk Learn, 2024)](https://www.zotero.org/google-docs/?HFnEhu)

A04:2021 - Insecure Design

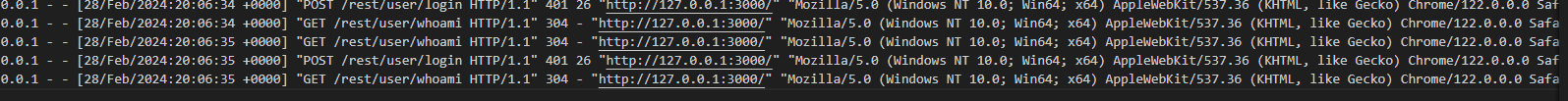
Problem:

This is when security aspects of an application are not considered when designing and implementing said application. An application can be implemented correctly as designed with no human implemented errors present in said application, but if the application is not designed properly it can still be vulnerable. An example of an insecure design would be connecting to the web without installing a firewall. A firewall in fundamental when connecting to the web as it protects from outside sources but if the application wasn’t intended to run with a firewall it would mean that while the application was implemented correctly, it was designed incorrectly which is where the vulnerability is showcased [(OWASP, 2021d)](https://www.zotero.org/google-docs/?bNRTLA)

Forces:

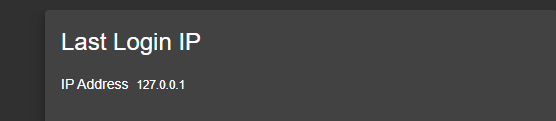
In this application we can see two areas were poor design choices are causing a vulnerability

1st we can see that the application does not prevent multiple login attempts from users. A user could attempt to log into an account 100 times and get it wrong but the application does not have any system in place to make it more secure. This means that the application is potentially vulnerable to brute force attacks due to no system being implemented after multiple attempts of a failed login



# (Screenshots of the access logs. It shows a login attempt being made but doesnt show it failed and/or what user was being logged in. "G:\juice\_shop\juice-shop\logs\access.log.2024-02-28")

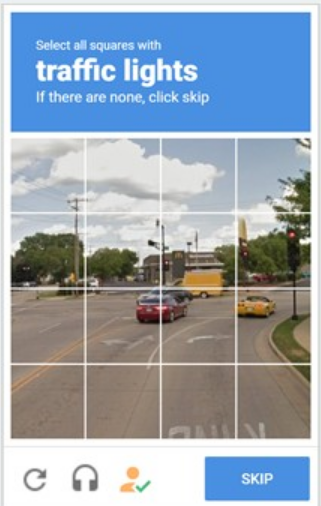
2nd we can see the application showcases the ip address of the user. This is sensitive information that can be used in potentially harmful ways. This info should not be showcased so easily and should be kept private. This information can be located by clicking on the Account > Privacy & Security > Last Login Ip . If anyone was listening in onto the users computer/network they could access the Ip address easily



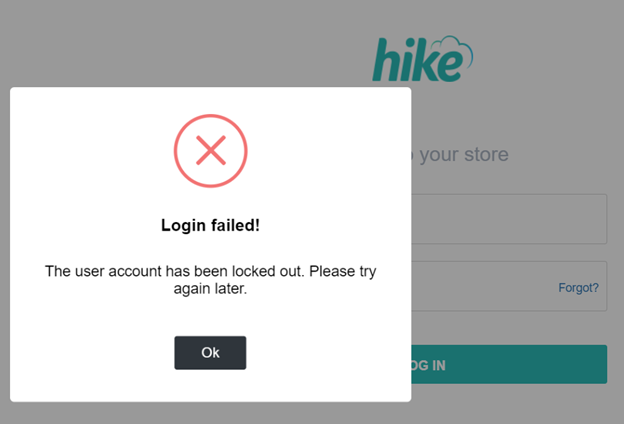
# (Screenshot from the application showcasing ip address.)

Solution:

1. For the first issue, there are multiple different methods we can use to fix this. We have already mentioned 2FA above, which would prevent the wrong user from accessing the account so we won't reiterate this but it is a potential solution.
2. After a set amount of failed login attempts the application could start requesting more from the user before they can attempt to log in. Recently this is normally done through the use of Captcha as it is randomised making it harder for automated attacks to get through the security

[](https://addons.opera.com/en-gb/extensions/details/buster-captcha-solver-for-humans/)

(Example of a Captcha [(Opera Add-Ons, 2023)](https://www.zotero.org/google-docs/?bjutG7) )

1. Another option that isn't as commonly used in recent times is a timed lock before the user can attempt to log in again. Again this helps minimise the dangers of brute force attacks on the application as it can greatly reduce the speed in which the attacks can happen. If using this method ensure that it is implemented correctly on the server side as if its a wrong implementation for example implemented on client side, this can be bypassed [](https://help.hikeup.com/portal/en/kb/articles/what-to-do-if-my-account-is-locked)

# (Example of a timed lock out [(Hike, 2023)](https://www.zotero.org/google-docs/?UlNshl))

In terms of the Ip address being showcased a simple solution of just remove this should suffice

Consequences:

By implementing the changes such as captcha ,2FA as well as timed lock out of logging in, it should make it incredibly more difficult to brute force attack the application to be able to log in to other users accounts.

By also not publicly displaying the ip address it increases the safety to the user as their location and details can be located through this information

Related Aspects:

A secure design by current standards can become outdated in the future. To keep the application designed in a secure way, the client needs to keep updated on current security practices. As different packages and software can become outdated, so can security designs.

Owasp also has a list to keep in mind for secure design practices. These are guidelines for when creating an application to consider to help keep secure practices in mind. An example of some of these security topics are “least privilege” and “fail safe”.

Least privilege states that users should be given the minimum amount of permissions that are necessary for them to complete their tasks along with a time limit on these permissions. The time limit should be implemented to prevent the escalation of roles/permissions

Fail safe states that the application needs to maintain its confidentiality, integrity and also its availability when an error message is triggered. Error messages can cause applications to become vulnerable. Usually this is implemented by having a secure fail state if this should happen. [(OWASP, 2024a, 2024b)](https://www.zotero.org/google-docs/?jRZVsn)

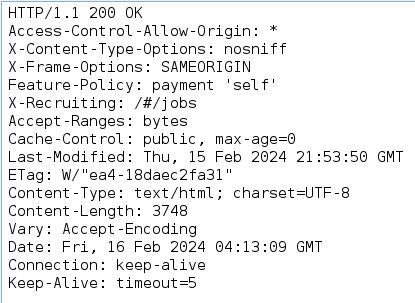
[(OWASP, 2021d)](https://www.zotero.org/google-docs/?JQqVAx)

A05:2021 - Security Misconfiguration

Problem:

When software, servers, or web applications are delivered with unsafe configurations or default settings, misconfiguration, a common security flaw, occurs. These configuration errors put systems open to many kinds of attacks and exploitation, compromising the availability, confidentiality, and integrity of data and services. By using scanning tools (ZAP Proxy) we were able to detect listed vulnerabilities [(Zap Dev Team, 2024)](https://www.zotero.org/google-docs/?glfgIt).

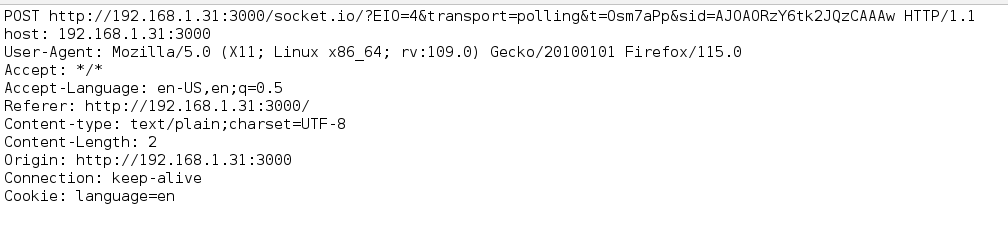
1. Content Security Policy (CSP) header not set

****

# (Screenshot from ZAP showcasing header not being set)

[(Caniuese.com, 2024; Content Security Policy (CSP), 2023; Medley and West, 2024; Mozilla Developer, 2024; Owasp Cheat Sheet Series, 2024; w3.org et al., 2024)](https://www.zotero.org/google-docs/?d9kjSt)

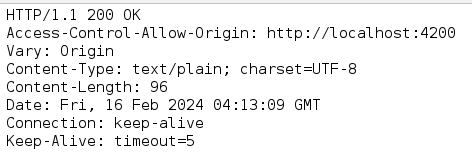
2. Missing Anti-clickjacking Header

****

# (Screenshot from ZAP showcasing header not updated with Anti-clickjacking setup)

[(Mozilla Developer, 2023)](https://www.zotero.org/google-docs/?cVRyrC)

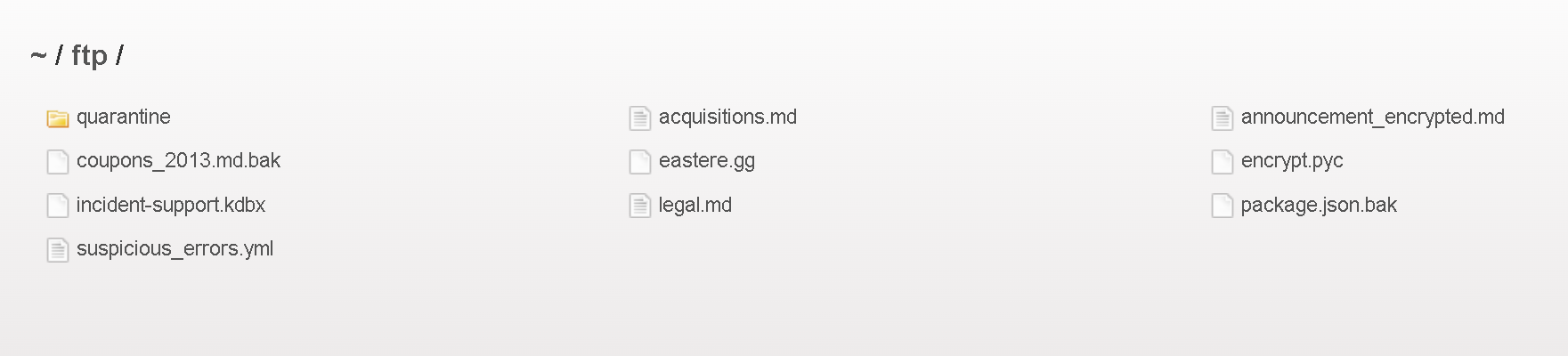
3. X-Content-Type-Options Header Missing

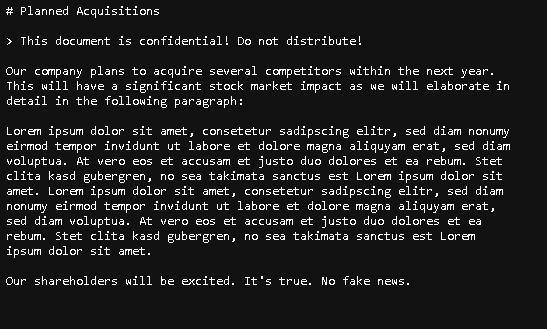
****

# (Screenshot from ZAP showcasing missing X-Content-Type-Options Header )

[**(Albeniz et al., 2023; Microsoft Learn, 2016)**](https://www.zotero.org/google-docs/?svCtn0)

4. Directory listing

****

****

# (Screenshot from application showcasing directories that can be accessed by attackers for sensitive information)

[(Common Weakness Enumeration, 2024a)](https://www.zotero.org/google-docs/?U8UrL1)

Forces:

1. These attacks are used for a variety of purposes, including distribution of malware, site alteration, and data theft.

2. By embedding a malicious website in a frame or iframe, a malicious website can hijack clicks meant for another website. This form of attack is known as clickjacking.

3. Not configured to 'nosniff' in the Anti-MIME-Sniffing header X-Content-Type-Options. Due to this, the response body may be interpreted and displayed as a content type different from the declared content type in older versions of Internet Explorer and Chrome when MIME-sniffing is performed on it. Since error type pages (401, 403, 500, etc.) are frequently still impacted by injection issues, browsers sniffing those pages away from their actual content type is still a worry. This issue therefore still applies to those sites.This scan rule will not notify the user of client or server error answers when the threshold is set to "High".

4. The term "directory listing" describes a web server's capacity to show directory contents in the absence of a default file (like index.html). When private material, including configuration files or user information, is unintentionally made available to users or possible attackers, there is a risk of directory listing.

Solutions:

1. Make that the Content-Security-Policy header is set on your web server, application server, load balancer, etc. through configuration.

2. The X-Frame-Options and Content-Security-Policy HTTP headers are supported by most modern web browsers. Need to make sure that one of them is selected on every webpage that is returned by site or app. Using SAMEORIGIN if anticipating that the page will only ever be framed by pages on your server (for example, if it is a member of a FRAMESET). If not, should use DENY. An alternative is to implement Content Security Policy's "frame-ancestors".

3. Making sure the application or web server sets the X-Content-Type-Options header to 'nosniff' for every web page and that the Content-Type header is set appropriately. Make every effort to guarantee that the end user is using a current, standards-compliant web browser that either doesn't do MIME-sniffing at all or can be instructed by the web application/web server to do so.

4. Directory listing can be stopped by configuring a web server to not allow it. Most of the time, this can be accomplished by making changes to the configuration file (such as nginx.conf or httpd.conf) from a server.

Consequences:

1. An additional layer of protection called Content Security Policy (CSP) aids in the detection and mitigation of some attack types, such as data injection and cross-site scripting (XSS) attacks. With the help of a set of standard HTTP headers called CSP, website owners can specify which content sources—JavaScript, CSS, HTML frames, fonts, images, and embeddable objects like Java applets, ActiveX, audio, and video files—browsers are permitted to load on their page.

2. Implementing anti-clickjacking headers enhances security by preventing malicious websites from embedding site pages, safeguarding sensitive data and ensuring compliance with security standards.

3. Setting "no sniff" on every webpage prevents browsers from performing MIME type sniffing, reducing the risk of certain types of attacks, such as cross-site scripting (XSS), by ensuring that content is interpreted as intended without any unexpected behaviour.

4. Configuring changes to stop directory listing enhances security by preventing unintended exposure of sensitive information and directories on your web server, reducing the risk of unauthorised access and potential exploitation by attackers.

Related Aspects:

1. The use of nonces and hashes for inline resources, browser compatibility, policy granularity, reporting and monitoring mechanisms, and keeping up with any changes or advancements in CSP technology are all important factors to take into account when dealing with the lack of Content Security Policy (CSP) headers. These elements support efficient security while preserving the functionality of applications and adjusting to changing web standards.

* To manage the framing behaviour, we use the X-Frame-Options header.
* For further control over framing, we use Content Security Policy (CSP) in conjunction with the frame-ancestors directive.
* In order to control information leaking, we need to set the Referrer-Policy header.
* Need to be aware of new developments in technology, such as Feature Policy, to ensure future improvements.
* For security advice, consulting resources from groups such as OWASP.
* Monitoring updates to browser specs and web standards.
* We need to Update and review security headers on a regular basis.
* Performing regular security audits in order to find and fix issues.



* Adding the value "nosniff" in the X-Content-Type-Options header to stop MIME type sniffing.
* To further improve security, employing additional security headers like Content Security Policy (CSP).
* Need to be updated on standards and best practices for online security.
* Reviewing and updating security headers frequently to keep up with emerging risks.
* For security advice, consulting resources from groups such as OWASP.
* Security audits are a proactive way to find and fix vulnerabilities.
* Disabling directory listing on the web server to stop unwanted users from accessing the contents of the directory.
* Setting up the proper file permissions to limit access to private folders and files.
* To stop malicious queries, putting security measures in place like Web Application Firewalls (WAFs).
* Using online security technologies to frequently search for and fix vulnerabilities related to directory listings.
* Updating plugins and modules for web server software to minimise known vulnerabilities.
* Keeping up with new threats and technological advancements in web security to adjust defences appropriately.
* To find and fix vulnerabilities early on, conduct regular security audits.
* For security advice, consulting resources from groups such as OWASP.

[(OWASP, 2021e)](https://www.zotero.org/google-docs/?rMAb0D)

A06:2021 - Vulnerable and Outdated Components

Components that are out of date or vulnerable are among the most common security issues that software systems encounter nowadays. In order to add more functionality or features to software projects, these components can be libraries, frameworks, plugins, or other third-party dependencies. If not properly controlled, these components pose serious security threats even if they can speed up development and improve functionality [(OWASP, 2021f)](https://www.zotero.org/google-docs/?gpQ02U).

Problem:

1. Outdated packages

# (List of outdated packages scanned by using Snyk)

Forces:

1. The risk of outdated packages includes vulnerabilities, compatibility issues, and potential security breaches, jeopardising the stability and security of a software application.

Solutions:

1. Update outdated packages. To automate the setup of systems and apps, use a powerful configuration management solution. These technologies make it possible to impose security setups in a variety of situations and manage them remotely.

Consequences:

1. Ultimately, updating out-of-date packages strengthens the functionality and dependability of a software program by enhancing security, enhancing performance, ensuring compatibility, and lowering the danger of vulnerabilities.

Related Aspects:

* Often upgrading software packages to strengthen security and repair known flaws.
* Automated updates, using package managers that have updating methods integrated in them.
* To effectively track and manage dependencies, using version management systems.
* Keeping an eye on software developers and security communities updates regarding security advises and alerts.
* When considering which upgrades to prioritise, we need to use vulnerability detection tools.
* To isolate apps and their dependencies, by using virtualization or containerization.
* Following best practices when developing and deploying software to ensure security.
* To find and fix vulnerabilities, frequent penetration tests and security audits are necessary.

[(OWASP, 2021f)](https://www.zotero.org/google-docs/?bfwsXM)

A07:2021 - Identification and Authentication Failures

Problem:

This is when the application fails to confirm the identity of the user or fails to securely authenticate them, allowing for incorrect users to be able to access the application. This section relates to previously spoken about issues, such as A01:2021 – Broken Access Control and Cryptographic Failures. By having a weakness in the authentication of the application, the accounts become jeopardised on the platform [(OWASP, 2021g)](https://www.zotero.org/google-docs/?6WtD87).

Forces:

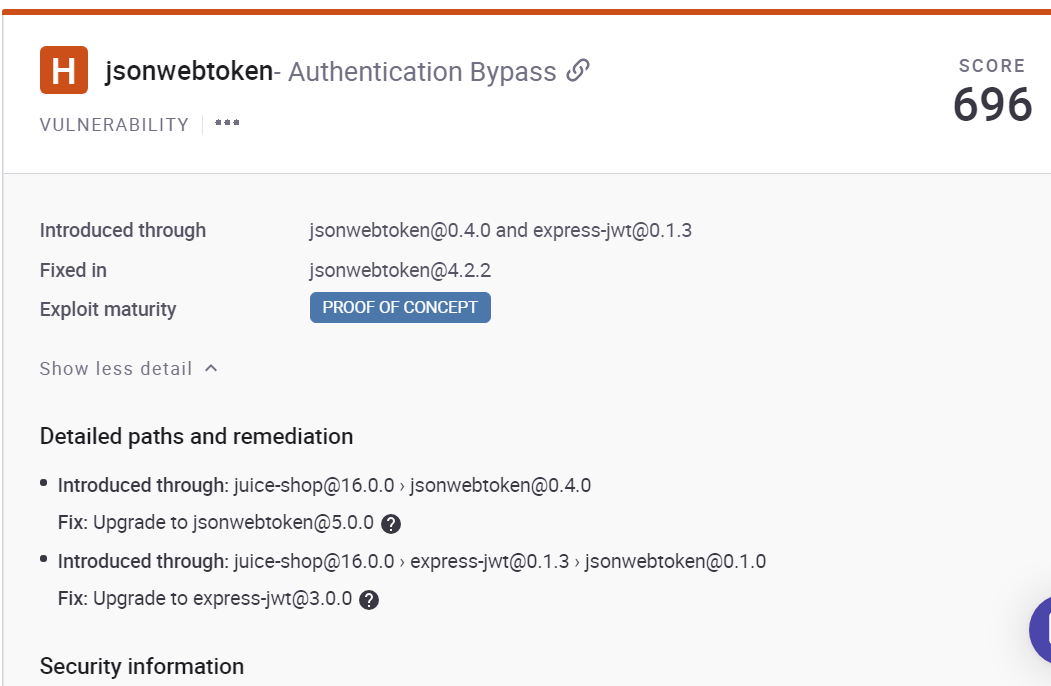
A Lot of the potential causes are present in this application. We have also addressed some potential solutions to these issues also. Due to this, reoccurring issues will just be highlighted and directed to where a deeper explanation was presented

This application does not enforce 2FA and does not confirm the email address of users before creating an account. This also allows malicious users to access/create accounts without confirming their identity. This issue was addressed in A01:2021 – Broken Access Control section with potential solution also advised

The application also allows users to create accounts with weak passwords, allowing accounts to be easily accessed as the password policy is not enforced. This issue was also mentioned in A01:2021 – Broken Access Control section

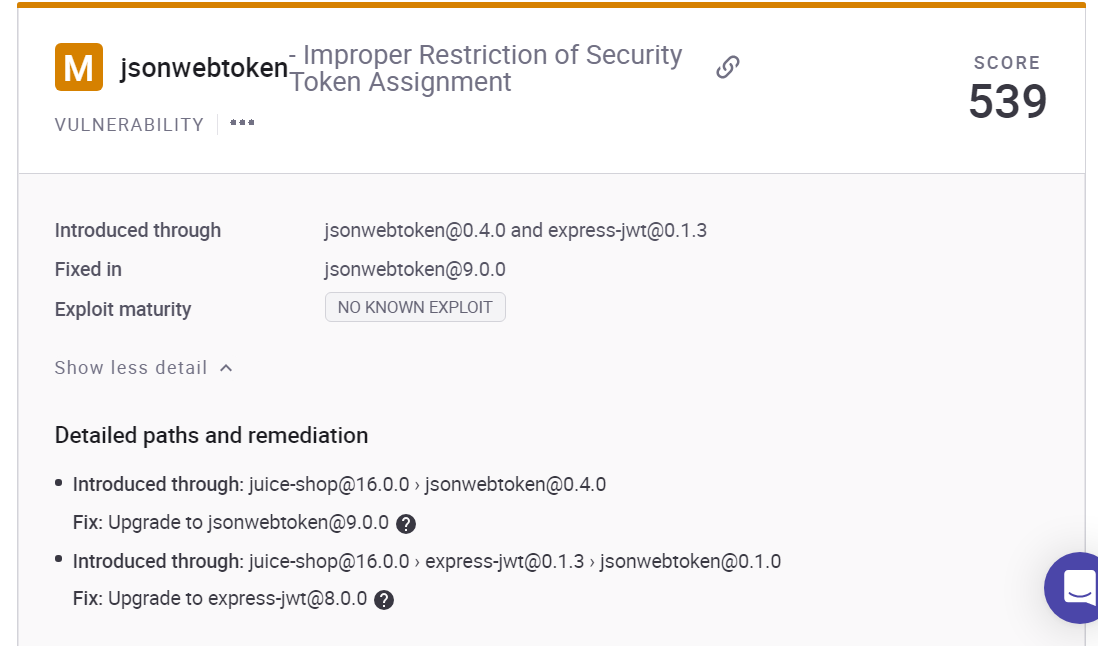
It doesn't seem that the application logs login attempts in great detail which could allow for a brute force attack to happen unrecognised. This is talked about in A09:2021 – Security Logging and Monitoring Failures and briefly mentioned in A04:2021 – Insecure Design.

In this application we can see that there is an issue with the authentication tokens. These authentication tokens are used to authenticate the user once they have already logged in so credentials don't have to be re-entered every time [(Okta, 2024)](https://www.zotero.org/google-docs/?T4WupV). This application has multiple vulnerabilities that stem from issues with these tokens



# (Screenshot from Snyk showcasing that there is an authentication bypass vulnerability)

According to the Snyk, it mentions that this vulnerability is caused by the authentication algorithm not being enforced. Due to this it allows attackers to bypass this by generating random signatures and keys.



# (Screenshot from Snyk showcasing there's another vulnerability with the security token)

This vulnerability is caused by the same packages showcased above but how the vulnerability is utilised is slightly different. It mentioned that there is an issue with the package verifying the keys incorrectly. It can potentially verify an asymmetric public key with a symmetric HS256 algorithm [(Johnson et al., 2022)](https://www.zotero.org/google-docs/?hYxSuX). This affects the security and the integrity of the application.

Solution:

Besides the issues mentioned above that have been addressed in other sections, the authentication bypass issues as well as the security token issue. Snyk mentions that both of these issues can be resolved by updating the associated packages. In these cases the vulnerable packages are jsonwebtoken [(auth0, 2024)](https://www.zotero.org/google-docs/?BLkC4X) and express-jwt [(auth0, 2023)](https://www.zotero.org/google-docs/?cQVtnA). The vulnerabilities exploit the security issues known in these packages hence an update was provided to rectify this issue. By updating the packages the application will become more secure against these types of attacks.

Consequences:

By implementing the solution it strengthens its resistance to these authentication attacks as it becomes more secure against these known types of attacks. The known vulnerabilities become patched meaning attacker will need to utilise another method to exploit the application

Related Aspects:

With computers getting faster and more capable as well as the introduction of quantum computing, security measures that are currently being utilised can quickly become outdated. It is recommended that the client keeps all packages up-to-date to prevent exploits from affecting the applications. Clients should also ensure that they keep up with current security practices to ensure that applications stay secure. As already mentioned, the use of a security development platform can help automate aspects of security for the client to assist in keeping up with the security concerns of the application [(OWASP, 2021g)](https://www.zotero.org/google-docs/?VeH1OQ)

A08:2021 - Software and Data Integrity Failures

The accuracy, reliability, and trustworthiness of software programs and the data they handle are put at risk by software and data integrity failures, which are serious security incidents. These errors arise from unapproved additions, deletions, or corruption that compromises the accuracy of databases, configuration files, software code, or other vital resources. Data loss, system instability, unauthorised access, and confidentiality breaches are just a few of the serious outcomes that can result from software and data integrity errors [(OWASP, 2021h)](https://www.zotero.org/google-docs/?woYBGb).

Problem:

1. Cross-Domain JavaScript Source File Inclusion

****

# (Screenshot from ZAP showcasing Cross-Domain JavaScript Source File Inclusion )

[(Common Weakness Enumeration, 2024b)](https://www.zotero.org/google-docs/?Op1TUp)

Forces:

1. A security flaw known as "cross-domain JavaScript Source File Inclusion" occurs when a webpage contains a JavaScript file from another domain. This flaw could provide hackers the ability to steal information, take over sessions, run malicious code, send malware, carry out phishing attacks, or deface websites. Threats like malware distribution, phishing assaults, data theft, session hijacking, cross-site scripting (XSS), and website defacement might result from it.

Solution:

1. Server-side procedures, Content Security Policy (CSP) headers, secure coding techniques, and frequent security audits are examples of mitigation techniques. Making sure the application's end users cannot influence the sources from which JavaScript source files are loaded.

Consequences:

1. By efficiently managing the sources from which JavaScript files are loaded, the application may lower security risks and improve overall defence against a variety of cyberthreats.

Related Aspects:

* The domains that are allowed to load JavaScript files might be restricted using Content Security Policy (CSP) headers.
* To stop injection attacks, server-side validation and user input sanitization implementation is necessary .
* Reducing the number of vulnerabilities in web apps by using secure coding techniques.
* Updating and patching software often to address known security flaws.
* Making sure that external script files are intact, we need to utilise Subresource Integrity (SRI).
* Keeping an eye on and checking online traffic for any unusual behaviour or illegal access.
* Keeping with the latest developments in online security for best practices and threats.
* Working together with communities and security researchers to find and fix new vulnerabilities.

[(OWASP, 2021h)](https://www.zotero.org/google-docs/?0fY7Tp)

A09:2021 - Security Logging and Monitoring Failures

Problem:

Robust cybersecurity strategies must include effective security logging and monitoring because they offer insight into system operations, identify suspicious activity or security issues, and facilitate prompt reaction and remedial steps. However, there are a number of serious hazards and difficulties associated with current security logging and monitoring setup, such as:

· Lack of Visibility: Inadequate recording and monitoring procedures lead to a lack of insight into system operations, which makes it challenging to identify and appropriately handle security risks or events.

· Missed Threats: There is a greater chance of ignoring security threats or suspicious activity in the absence of thorough recording and monitoring capabilities, which raises the possibility of security breaches or data compromises.

· Compliance Concerns: A company may face legal and financial repercussions if its security logging and monitoring procedures are inadequate and lead to non-compliance with industry or regulatory regulations.

· Delayed Response: Security events may remain undetected for long stretches of time in the absence of prompt detection and alerting methods, giving attackers more time to stay in the system and do more harm.

[(OWASP, 2021i)](https://www.zotero.org/google-docs/?fBti08)

Forces:

The application does not appear to log/monitor login attempts and doesn't take any action when multiple failed attempts have been made. Due to this it can make the application vulnerable to brute force attacks as there is no limit to the amount of attempts that can be made.

Solution:

· Implement Comprehensive Logging: Install reliable logging methods for all important applications, systems, and network infrastructure parts. Make sure that all pertinent system activity, security events, and access logs are recorded and collected in one place for analysis.

· Deploy Advanced Monitoring Solutions: Invest in modern safety monitoring systems that may instantly identify possible threats or security problems by analysing log data in real-time, spotting abnormalities, and connecting security occurrences.

· Define Clear Alerting Criteria: Establish precise standards for producing warnings that are triggered by known indicators of compromise (IOCs), suspicious activity, or infractions of security policies. Set up alerting systems so that when possible security incidents are discovered, security teams or incident response staff are notified right away.

· Implement Continuous Monitoring: Manage networks, systems, and apps continually by putting continuous monitoring procedures into place. To find and proactively fix vulnerabilities or new threats, thoroughly examine and analyse log data on a regular basis. Engage in threat hunting exercises and security assessments.

· Enhance Incident Response Capabilities: To guarantee a prompt and well-coordinated response to security issues, develop and record incident response protocols. To evaluate the efficacy of response protocols and enhance incident response preparedness, conduct routine tabletop exercises and incident response drills.

Consequences:

It will strengthen capacity to identify, address, and mitigate security risks by putting the suggested security logging and monitoring methods into practice. This will lower the likelihood of security breaches and protect vital assets and data.

Related Aspects:

Real-time alerting systems to notify security teams of unusual activity, integration with threat intelligence feeds for proactive threat identification, and frequent security training for staff to improve knowledge of security logging and monitoring procedures are other factors to take into account.

For information on putting into practice efficient security logging and monitoring solutions, see industry best practices such as the OWASP Logging Cheat Sheet, CIS Controls, and NIST Cybersecurity Framework.

In the future, we expect to see developments in artificial intelligence and machine learning for automated anomaly detection and behaviour analysis; cloud-native logging and monitoring solutions being adopted for scalability and flexibility; and blockchain technology being integrated for tamper-proof logging and immutable audit trails [(OWASP, 2021i)](https://www.zotero.org/google-docs/?YlwUyb)**.**

A10:2021 – Server-Side Request Forgery (SSRF)

Problem:

Server-Side Request Forgery (SSRF) is a security vulnerability that occurs when an attacker is able to trick a server into making an unintended HTTP request to a resource controlled by the attacker [(OWASP, 2021j)](https://www.zotero.org/google-docs/?WK7J8T).

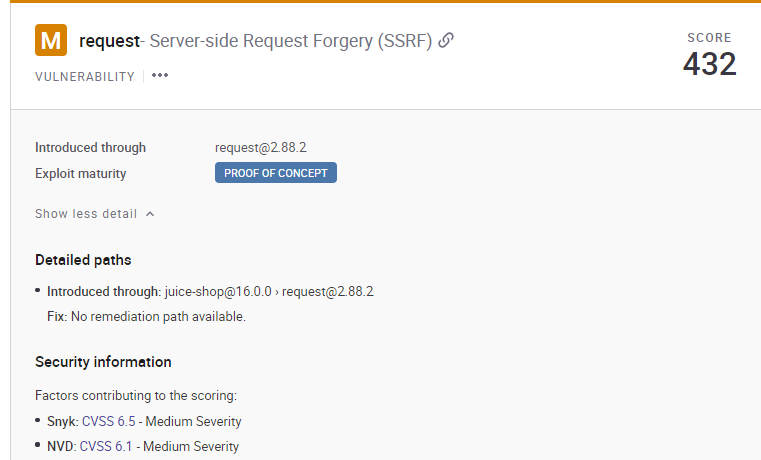
Forces:

Actors , technical constraints, and technological limitations play crucial roles in mitigating Server-Side Request Forgery (SSRF) vulnerabilities within an application. The actors involved in addressing SSRF vulnerabilities include developers, security analysts, and system administrators. Each has a specific responsibility in identifying, preventing, and resolving SSRF issues.

From a technical perspective, implementing input validation, enforcing secure coding practices, and utilising web application firewalls are effective measures to mitigate SSRF risk. Furthermore, leveraging modern technologies such as containerization and cloud-based security solutions can enhance the overall resilience of the application against SSRF attacks. By understanding and addressing these constraints, organisations can significantly reduce the impact of SSRF vulnerability.

Solution:

* Consider using a proxy server to control the outgoing requests from the server, limiting the servers to only make requests to trusted external resources.
* Implement proper network segmentation to restrict the communication channels between different parts of the application and internal resources.
* Disable any unnecessary services or functionalities that can be exploited through SSRF attacks.
* Ensure that sensitive internet resources are properly protected with strong access controls. Such as authentication and authorizations.

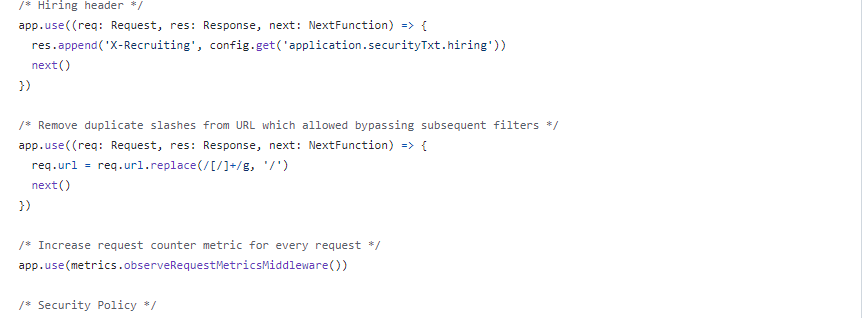


# (The screenshot above is taken from Snyk, highlight the affected version of a simplified http request client)

According to Snyk overview in the above screenshot:

[request](https://www.npmjs.com/package/request) is a simplified http request client.

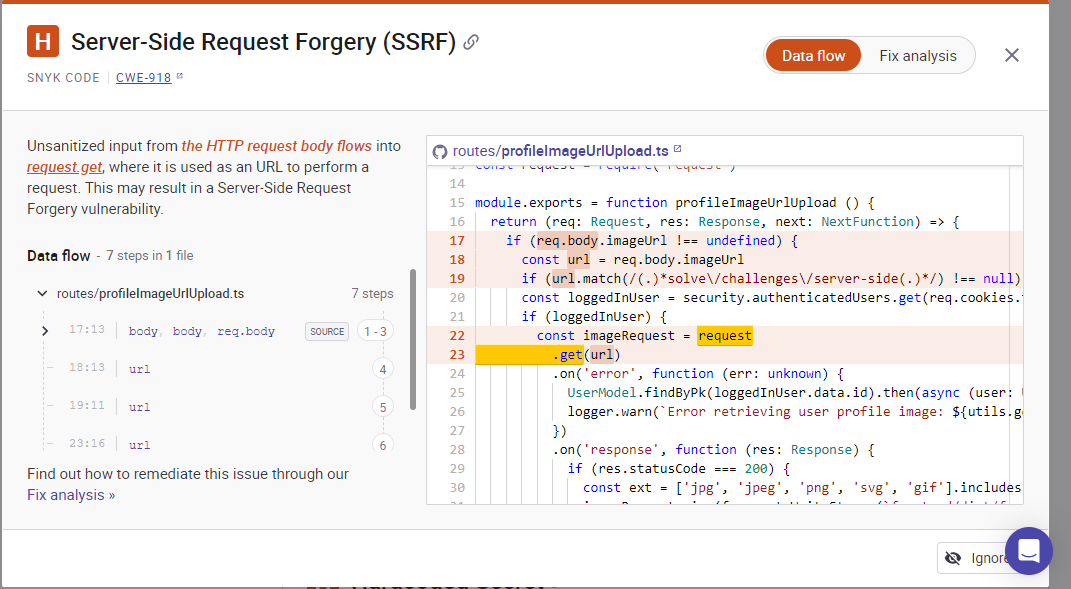
Affected versions of this package are vulnerable to Server-side Request Forgery (SSRF) due to insufficient checks in the lib/redirect.js file by allowing insecure redirects in the default configuration, via an attacker-controller server that does a cross-protocol redirect (HTTP to HTTPS, or HTTPS to HTTP).



# (The above screenshot from Juice-shop application revealed unvalidated and unsanitized inputs)

The above screenshot shows that the inputs are not properly validated or sanitised as the header accepts URLs which allows bypassing subsequent filters.

In any application or code, the user input for making HTTP requests such as fields, parameters, or headers should not accept URLs or hostnames to prevent SSRF vulnerability.



# (screenshot from Snyk revealed the server-side request vulnerability)

The above screenshot shows another server-side request forgery vulnerability. According to the scan done with the Snyk [(Snyk, 2024)](https://www.zotero.org/google-docs/?TqkyZ0), it reveals that in a server-side request forgery attack, a malicious user supplies a URL (an external URL or a network IP address such as 127.0.0.1) to the application's back end. The server then accesses the URL and shares its results, which may include sensitive information such as AWS metadata, internal configuration information, or database contents with the attacker. Because the request comes from the back end, it bypasses access controls, potentially exposing information the user does not have sufficient privileges to receive. The attacker can then exploit this information to gain access, modify the web application, or demand a ransom payment.

Best practices for prevention

* Blacklists are problematic and attackers have numerous ways to bypass them; ideally, use a whitelist of all permitted domains and IP addresses.
* Use authentication even within your own network to prevent exploitation of server-side requests.
* Implement zero trust and sanitise and validate all URL and header data returning to the server from the user. Strip invalid or suspect characters, then inspect to be certain it contains a valid and expected value.
* Ideally, avoid sending server requests based on user-provided data altogether.
* Ensure that you are not sending raw response bodies from the server directly to the client. Only deliver expected responses.
* Disable suspect and exploitable URL schemas. Common culprits include obscure and little-used schemas such as file://, dict://, ftp://, and gopher://.

Consequences:

* The solutions above will enhance the overall security posture by preventing attackers from manipulating the application’s internal requests and potentially, accessing sensitive information or resources.
* Addressing SSRF vulnerability helps in safeguarding the application’s integrity and reliability, reducing the risk of unauthorised data exfiltration or malicious activities.
* Fixing SSRF issues can improve the trust and confidence of users, leading to increased user satisfaction and loyalty.
* Resolving SSRF vulnerability is crucial for ensuring the security, integrity, and trustworthiness of the application.

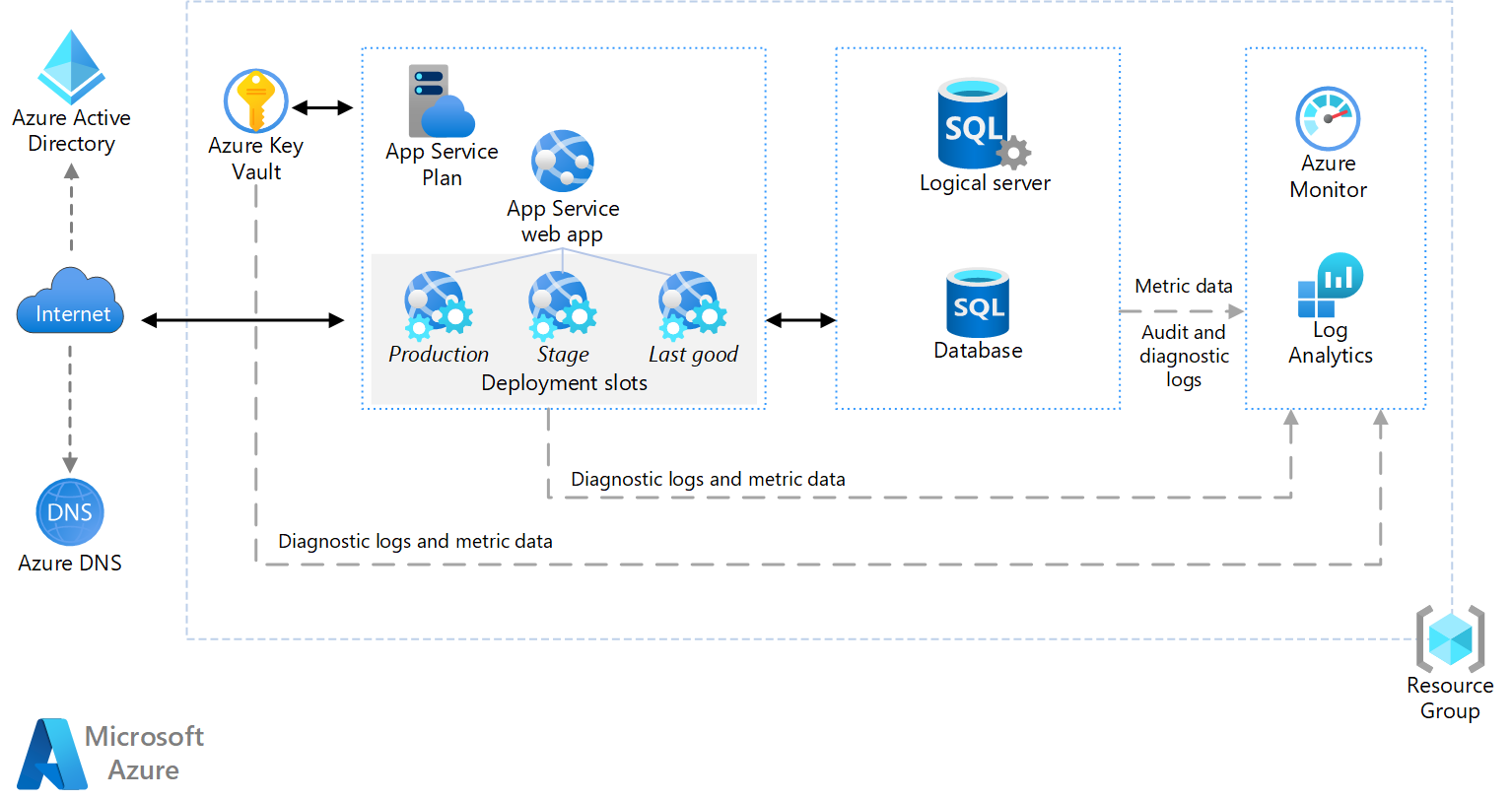
Related Aspects:

As technology advances, the consideration of SSRF vulnerabilities in applications becomes more critical. References to past incidents guide us in understanding the risk associated with SSR. Future changes in technology, such as enhanced input validation and secure coding practices, offer promising solutions to mitigate SSRF vulnerabilities effectively. As frameworks and languages evolve, developers must stay informed and adapt their strategies to address emerging threats. By prioritising security measures and adopting a proactive approach to development, we can anticipate and prevent SSRF vulnerabilities, ensuring the robustness and integrity of our applications in the ever-changing technological landscape.

[(OWASP, 2021j)](https://www.zotero.org/google-docs/?uAkHKC)

Proposed Solutions Architecture (Part 2)

In this section we research potential cloud service technologies that could be applicable to our application. After researching these technologies, we discuss which ones we deem necessary to the application in terms of security, while in turn keeping the topics of the CIA triad - Confidentiality, Integrity and Availability.

[](https://learn.microsoft.com/en-us/azure/architecture/web-apps/app-service/architectures/basic-web-app?tabs=cli)

# (Microsoft example of how the architecture of a web application hosted on azure should be structured [(Microsoft Learn, 2023b)](https://www.zotero.org/google-docs/?Ca6mrE) )

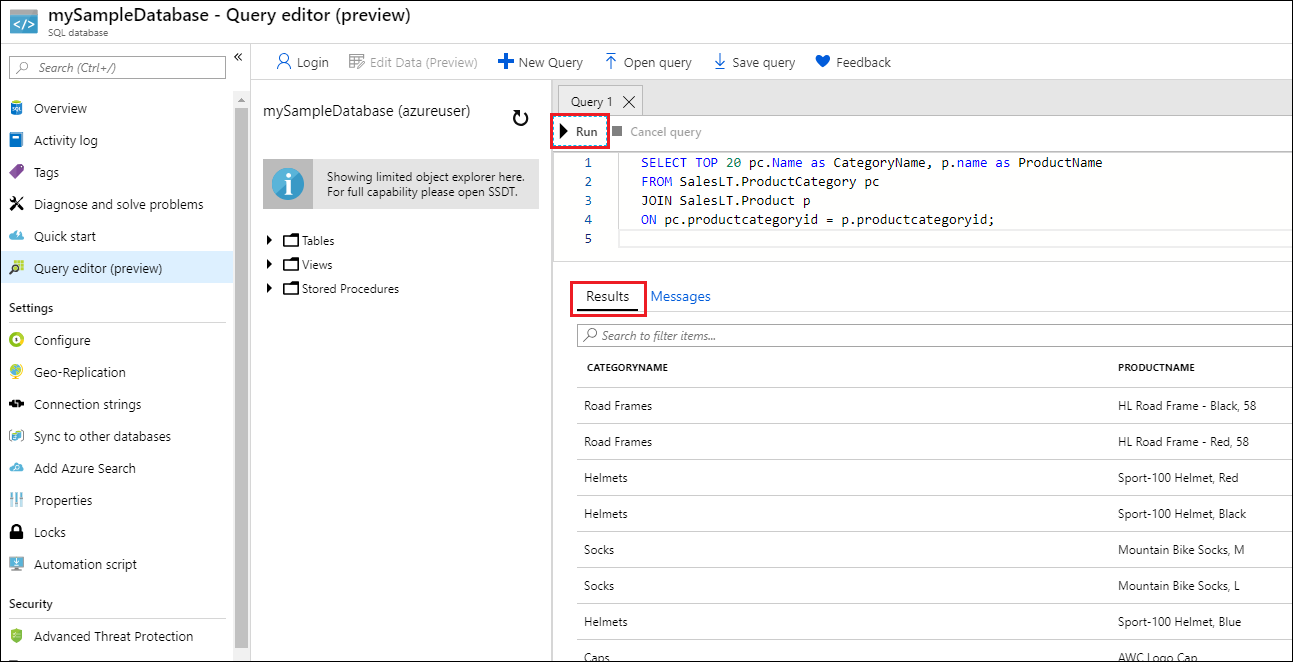
# 

# (Microsoft example of the structure a reliable web application should take [(Microsoft Learn, 2023c)](https://www.zotero.org/google-docs/?OJYrRL))

# 

SQL Databases

There are different options for database servers that we can use. We can use an oracle database, Azure SQL Database or Azure Database for MySQL. All are viable options but Azure SQL Database would be the ideal choice. Because we are hosting on an Azure server, it is easier to integrate and potentially manage the database if the service provided is the same. Microsoft defender for cloud would automatically pick up any Azure service and would automatically integrate them. Azure SQL Database is based on Microsoft’s SQL engine, due to this there would be a learning curve in learning the different engine. Due to it being Microsoft’s engine, this service does provide more features and better security compared to Azure Database for MySQL. Microsoft automatically keeps this updated due to it being a microsoft service so it doesnt need to be managed in the back-end . Azure Database for MySQL is an open source MySql database. This server is ideal for using MySql databases hence why this option is most ideal for us as the team currently has more experience with these types of databases. This service is automatically patched to keep service up to date and secure. Other aspects such as monitoring, backups as well as others can be managed manually or automatically by Azure. This service can also be used for the 1st 12 months for free, while it does have monthly limits it will allow us to test if we are happy with the service. This service can be used as IaaS if we need a hosted virtual machine infrastructure or PaaS if we need a hosted platform.[(Microsoft Learn, 2023d, 2024a, 2024b, 2024c; StackShare, 2023)](https://www.zotero.org/google-docs/?IAXXdV)

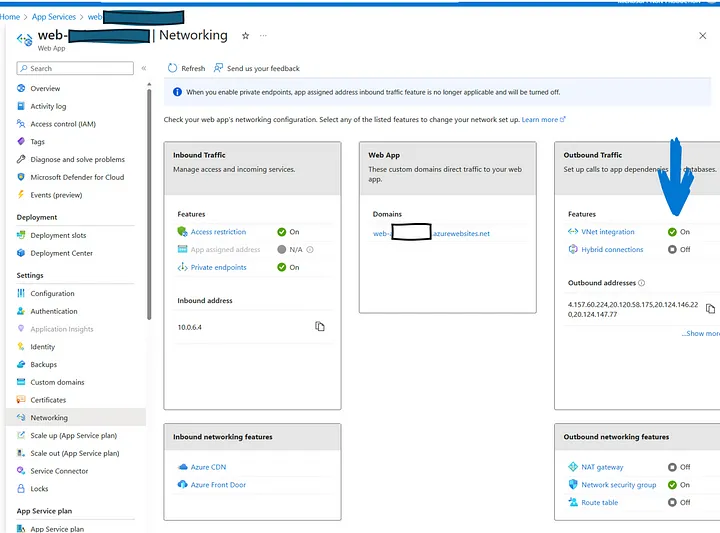
[****](https://learn.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?view=azuresql&tabs=azure-portal)

# (Sample image of how the UI looks for SQL Databases [(Microsoft Learn, 2024a)](https://www.zotero.org/google-docs/?gpp7EX))

Web App > Nodejs Server

Web Apps would be the ideal service for hosting the application. Web apps is an HTTP based service for hosting web applications. It is compatible with many languages such as Node.js which juice-shop is hosted in but also Python and Java to name a few. This service is highly compatible with Github and Visual studios which are industry standards and allow for easy integration with the service. Web Apps have built in security, load balancing and scaling. This allows Webs Apps to have a high uptime with 99.95% availability. This service is able to connect with **Microsoft defender for cloud,** **DevOps** and **Azure monitor** to name a few. Applications hosted on this service can be containerized and also connect to SaaS platforms such as Salesforce for an example. This service is also automatically updated to keep the service secure

This service works as a pay as you go service so users are charged for the compute resources they use.[(Edwards, 2021; Microsoft Azure, 2024a; Microsoft Learn, 2023e)](https://www.zotero.org/google-docs/?HnslZj)

[](https://jbrake30.medium.com/private-open-ai-service-in-azure-with-web-4ce0b70dd76c)

# (Example image of Web App’s UI [(Brakefield, 2023)](https://www.zotero.org/google-docs/?1qdJoN))

Ssh Keys - Azure Key Vaults

Due to the cryptographic failure detected as one of the vulnerabilities in Juice-shop application, **Secure Shell (SSH)** is highly recommended.

SSH keys are cryptographic keys used to securely authenticate users and establish secure connections over an insecure network. They come in pairs: a public key that is shared with servers and a private key that is kept secure on a server’s device. When a user tries to connect to a server, their client uses the private key to prove their identity, and the server uses the corresponding public key to verify it. This method provides a more secure way to authenticate users compared to passwords. SSH keys are widely used in various applications to ensure secure and encrypted communication over the internet.

**Azure Key Vault** is a secure cloud service offered by Microsoft Azure for storing and managing cryptographic keys, secrets, and certificates used by cloud applications and services. It provides a centralised location to safeguard sensitive information such as passwords, API Keys and secrets, monitor usage, and audit access. By utilising Azure Key Vault, businesses can enhance the security of their applications and effectively manage their encryption keys and secrets in a scalable and a reliable manner within the Azure cloud environment.

**Ref: Azure key vault documentation.**

Static Web Apps

Static web apps deployed using Azure containers offer numerous benefits to this application. By leveraging Azure containers, static web apps can achieve high scalability, improved performance, and enhanced security.

Static web apps offer enhanced security compared to their dynamic counterparts, Since there is no server-side processing involved, the attack surface is significantly reduced, making it harder for malicious actors to exploit vulnerabilities. Additionally, static web apps are easier to cache, further enhancing security by reducing the risk of data breaches.

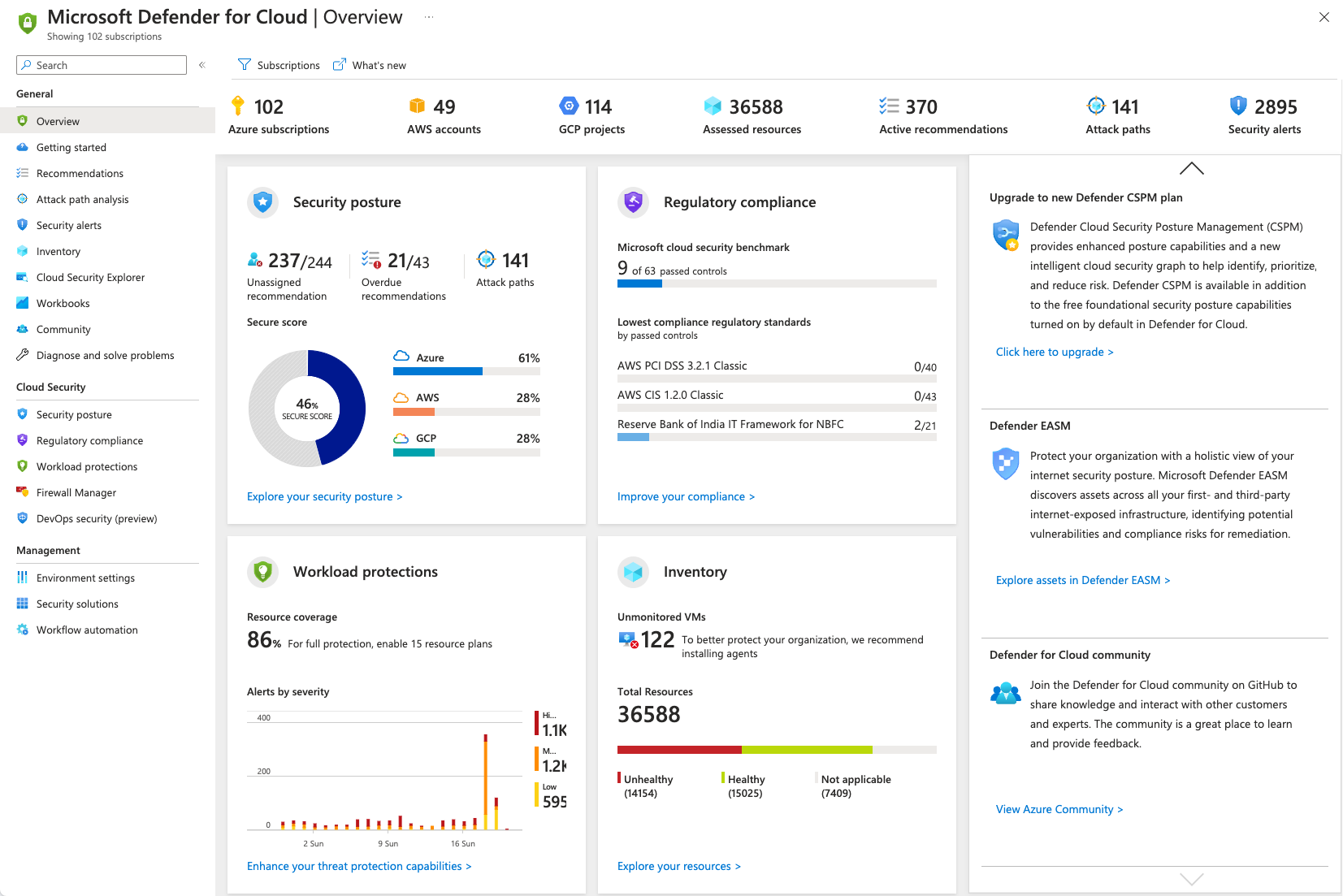
Another significant benefit using static web apps with Azure’s container to deploy this application is to simplify the process of scaling and updating the applications, leading to increased efficiency and reduced operational costs.

Furthermore, static web apps are highly cost-effective, as they require minimal server infrastructure and can handle high traffic volumes without incurring additional costs. This makes it an ideal choice for us to deploy and efficiently manage this application, ensuring consistency without incurring too much cost.

**Ref: Microsoft Azure’ official documentation on Azure container.**

Microsoft Defender For Cloud (Formerly Known As - Azure Security Center)

Security and auditing tool. It is designed for all users on the azure platform and it isn't designed for security professionals, allowing for ease of use. For example Secure Score is a grading scheme that allows users to analyse the security hygiene of their service. The score is given based on how well the service follows **Microsoft Best Practices** Scheme as well as the **Zero Trust Framework** [(Cloudflare, n.d.; Microsoft Learn, 2023f, 2023f, 2023g, 2024c; Microsoft Security, 2024)](https://www.zotero.org/google-docs/?m7RYOY). The service allows for insight into the security state of cloud workloads. It reduces exposure to attacks and allows for quick response to detected threats. Allows identification of missing security patches, missing/outdated anti-malware as well as insecure configurations of OS. It also provides insight into network,storage and data. Uses machine learning to help assist in blocking unnecessary applications and/or malware from running. Also assist in managing access to ports. When threats are detected, the Investigation path allows for easy investigation/understanding of the attack and areas that were compromised, this can then be used in conjunction with **Logic Apps** to deploy a resolution. Can also be used with **Sentinel** for hardened security (Cuff, 2021a, 2021b; Microsoft Azure, n.d.; Microsoft Learn, 2024, 2023a, 2023b, n.d.; Warner, 2023).

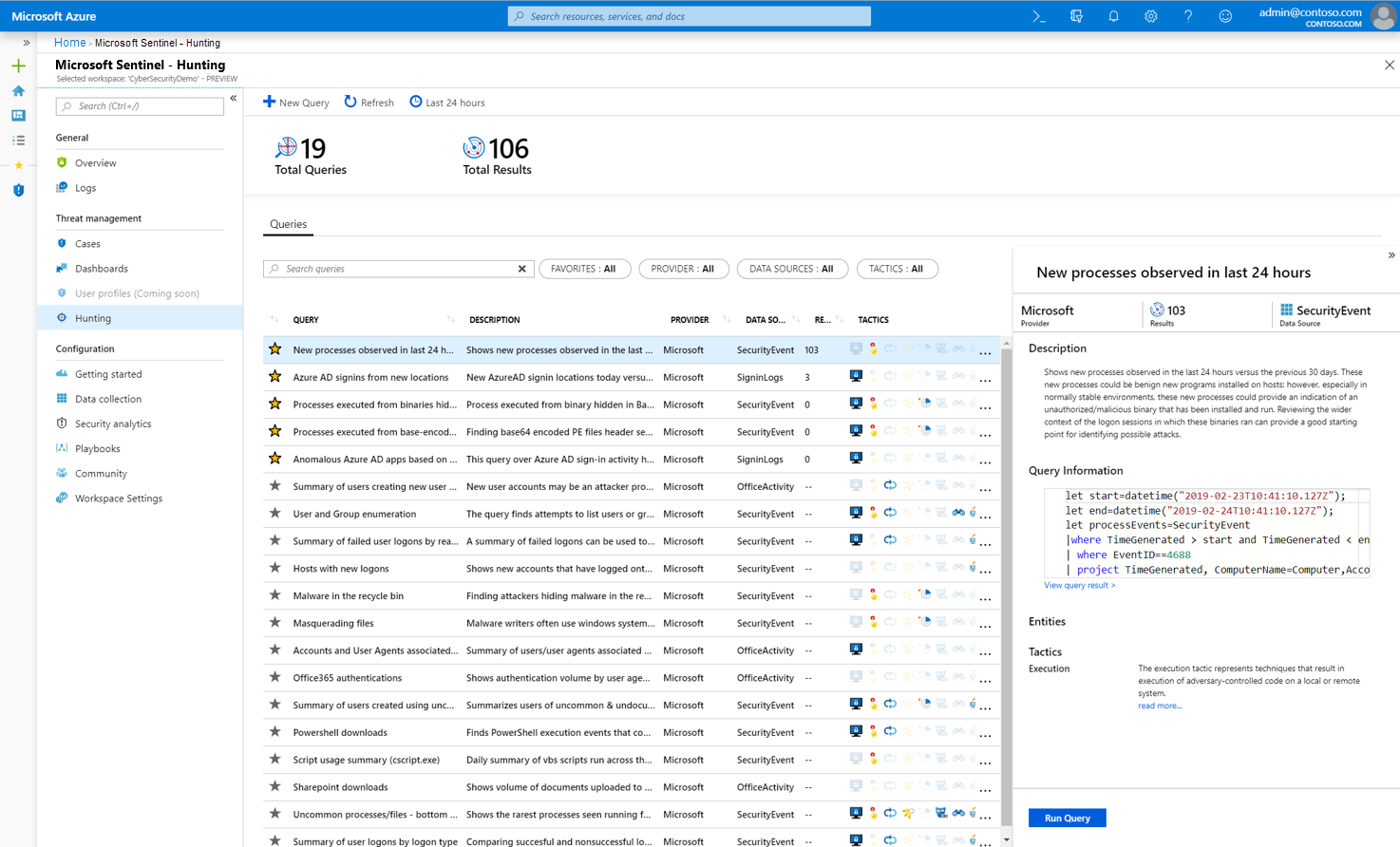
[](https://learn.microsoft.com/en-us/azure/defender-for-cloud/overview-page)

# (example of the UI used for Microsoft Defender For Cloud, this image also showcases the secure score aspect within the security posture tile [(Microsoft Learn, 2024d)](https://www.zotero.org/google-docs/?gC5MSH))

Microsoft Sentinel

Sentinel is designed to be used by a dedicated cloud security team. This service can also be used in tandem with **Microsoft Defender for Cloud** .Sentinel is a security information event management [(IBM, 2024a)](https://www.zotero.org/google-docs/?TUpxyw)(SIEM) and security orchestration automated response [(IBM, 2024b)](https://www.zotero.org/google-docs/?4MTLuR) (SOAR). A SIEM is a service that is used to recognise potential threats and vulnerabilities and rectify them before they can have a negative impact. These services tend to use artificial intelligence and machine learning to automate many of the processes needed for threat detection and response. A SOAR is a service that can group security alerts from multiple platforms into one centralised area and allows for an automated response for these detected threats [(IBM, 2024b, 2024b)](https://www.zotero.org/google-docs/?mu8qWB).

Sentinel can integrate with cloud services such as **Microsoft defender for cloud** but also 3rd party cloud services, such as Barracuda firewalls and Cisco firewalls which makes it ideal for a security team due to being able to analyse multiple aspects of the service[(Microsoft Learn, 2023a; Warner, 2023)](https://www.zotero.org/google-docs/?zYOfqa). Sentinel is designed to be a direct competitor to other SIEM/SOAR services such as Splunk Enterprise and LogRhythm [(Cuff, 2021; Microsoft Learn, 2023a; Warner, 2023)](https://www.zotero.org/google-docs/?Ajorl7)

[](https://learn.microsoft.com/en-us/azure/sentinel/overview)

# (Example image of Microsoft Sentinel UI [(Microsoft Learn, 2023a)](https://www.zotero.org/google-docs/?Iks0sl))

Dev Ops

Devops is a methodology that combines software development (Dev-Ops) with IT operations (Dev-Ops) to improve collaboration and efficiency in the software development lifecycle. It emphasises communication, automation, and integration between software developers and IT professionals. This approach aims to shorten the system development life cycle and provide continuous delivery of high-quality software. Dev-Ops practices include version control, continuous integration, automated testing, and continuous development. By implementing Dev-Ops, organisations can achieve faster time to market, increased development frequency, and improved product quality. Ultimately, Dev-Ops foster a culture of collaboration, innovation, and agility within development teams, leading to better software products and increased customer satisfaction.

Dev-OPs practices integrated with **Azure VMs** offer significant security. By automating security configurations and implementing infrastructure as code, It can ensure consistent security across environments.

Dev-OPs has the ability to quickly deploy security updates and patches through automated pipelines enhances the resilience of **Azure VMs** against cyber threats.

(Ref: Microsoft’s official documentation on Azure DevOps.2024)

Azure Functions

Azure Functions is a serverless computing server provided by Microsoft Azure that allows developers to run event-driven code without the need to manage infrastructure. With Azure Functions, developers can focus on writing small pieces of code that respond to various triggers such as HTTP requests, timers, queues, and more. This allows for a highly scalable and cost-effective solution, as users only pay for the resources their functions consume. Azure Functions support various programming languages like C++/ python, Javascript, and more, making it versatile for different types of applications. Azure Functions provide a flexible and efficient way to build and deploy applications in the cloud.

(Microsoft Azure 2024)

**Logic Apps**

Logic Apps is a key component of **Microsoft Azure’s integration services**, revolutionising the way businesses automate workflows and streamline processes. By providing a visual designer and a vast array of connectors, Logic Apps empower users to create scalable and efficient workflows without the need for extensive coding knowledge.

The combination of Logic Apps with **Azure Virtual Machines (VM)** offers a robust security framework with distinct advantages. Logic Apps enable the seamless automation of workflows, reducing manual intervention and associated security risks. By integrating Logic Apps with Azure VM, organisations can leverage advanced security features such as access control, network isolation, and encryption to fortify their systems against cyber threats effectively. This integration enhances security posture and promotes a resilient cloud infrastructure.

[(Microsoft Azure, 2024b)](https://www.zotero.org/google-docs/?T7IOhq)

Disk Encryption Options

There are different types of encryption available for the server that we can utilise. By default **Azure Disk Storage Server-Side Encryption** is enabled.

**Azure Disk Storage Server-Side Encryption**:

This service acts as the baseline to the encryption of the servers data and the other services that can be utilised are compounded onto this. This service encrypts data stored on Azure managed disks for OS and data at rest when on the Storage Clusters (encryption at host instead encrypts the data on the server hosting the VM before transmitting the data through to the **Storage Clusters** [(Microsoft Learn, 2024e)](https://www.zotero.org/google-docs/?Kb8AwO) . This service can be configured with a Disk Encryption Set (DES) which allows it to manage keys in addition to the base services. Data that is stored and encrypted by Azure is encrypted using 256-bit AES encryption, which is a strong block cipher which is compliant with FIPS 140-2. This service is provided at no extra cost to the user and it does not affect the performance of the storage. The downside to this base service is that it doesn't encrypt temporary disk and disk caches [(Microsoft Learn, 2024e)](https://www.zotero.org/google-docs/?DGt3N1)

Encryption at host:

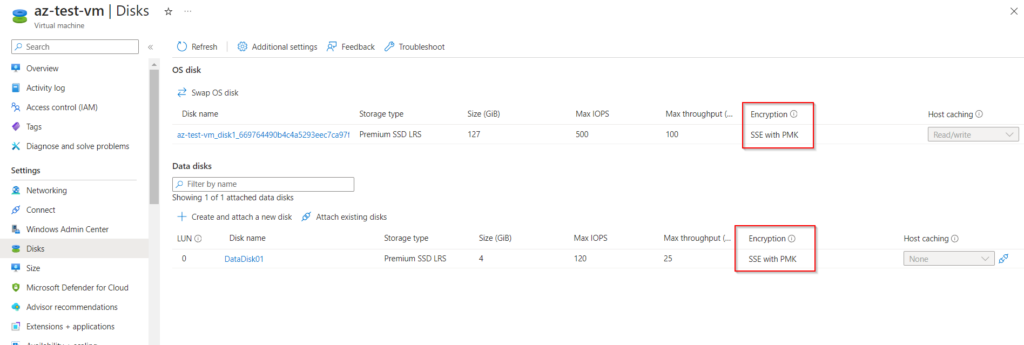
Is a service that compounds onto the base service. It's designed for virtual machines with the purpose of ensuring that temporary disks and disk caches are also encrypted for increased security. This is achieved by having the server hosting the virtual machine to encrypt all its associated data before transmitting data to the data clusters. The data stored at rest can be either encrypted with platform-managed keys or customer-managed keys. This is decided by how the user decides to configure their service. This service also does not affect the performance of the virtual machine and also can’t be used in conjunction with Azure Disk Encryption. [(Microsoft Learn, 2024f)](https://www.zotero.org/google-docs/?JVZGox)

Azure Disk Encryption:

Another option that is available is **Azure Disk Encryption**. It works by using BitLocker for Windows virtual machines or DM - Crypt for Linux virtual machines. This service works in conjunction with Azure Key Vault to help securely store associated keys. This service also provides encryption for temporary disk and disk caches and can also use customer managed keys. Unlike encryption at host, this service does use some compute resources due to using the built in encryption functionality of the Operating system.[(Microsoft Learn, 2024g)](https://www.zotero.org/google-docs/?dHZih0)

Conclusion:

Due to encryption at host being designed for virtual machines, and it doesn't use any compute resources, this service would be the ideal option for us to utilise. [(Microsoft Learn, 2023h, 2024e, 2024f)](https://www.zotero.org/google-docs/?rFJ66g)

[](https://www.azure365pro.com/deploy-azure-disk-encryption-for-data-disk/)

# (Example of a Server side encryption with Platform managed keys. This is showcased in the disk section of the virtual machine [(M, 2023)](https://www.zotero.org/google-docs/?azNrMG))

# (example of choosing different encryption option and key management systems [(Microsoft Learn, 2023h)](https://www.zotero.org/google-docs/?3Tqepw))

Zero Trust Framework

This framework means that everyone that attempts to connect to the resource/server is automatically not trusted until they can prove their identity, location as well as other factors that are deemed necessary before they are trusted to access the resource. Traditional IT network security trusts anyone and any connection that is within the network, this can allow for vulnerabilities, for example when an unauthorised user gains access to a physical location that allows them to access the network. By not trusting within and outside the network until identity has been verified, it's been shown that this added layer of security helps to reduce breaches. Other factors that are important to this framework are multi-factor authentication, monitoring devices and access as well as only providing the minimum level privileges so that the user can carry out their tasks.[(Cloudflare, n.d.; Microsoft Security, 2024)](https://www.zotero.org/google-docs/?4YVk2S)

Microsoft Best Practices

These are steps that Microsoft suggests that users take while designing, deploying and managing cloud solutions. They are intended for IT professionals to use but Microsoft defender for cloud as best practice built in that makes suggestions for users to quickly strengthen the security of the server. There are multiple sections for best practices that users can follow such as VM security best practices, PaaS best practices and database security best practises to name a few. By following these steps microsoft believes that the cloud service will be designed as ideally as possible for security[(Microsoft Learn, 2023f, 2023g)](https://www.zotero.org/google-docs/?oUfYew)

Azure VM

Applications, services, and workloads can be hosted in the cloud with Azure Virtual Machines (VMs), which offer a scalable and adaptable solution.

· Flexibility: With the variety of virtual machine sizes, operating systems, and settings that Azure VM offers, users can select the resources that best suit their needs. You can install Windows or Linux virtual machines (VMs) and modify them according to the requirements of your application.

· Scalability: You may scale your computer resources up or down in response to demand using Azure VM. To manage workload variations and maximise resource usage, you can add or delete instances, scale automatically, and modify the virtual machine (VM) size dynamically with Azure Auto Scaling.

· High availability: Features for guaranteeing high availability and dependability of your applications are integrated into Azure VM. In order to disperse workloads and reduce downtime in the event of hardware failures or maintenance events, virtual machine instances can be deployed across different Azure Availability Zones or Availability Sets.

· Integration: Azure VM easily connects with other services like Azure Security Center for threat detection and vulnerability management, Azure Monitor for monitoring and logging, Azure Storage for storing data and backups and Azure Virtual Network for network isolation and security.

· Security: Strong security protections are provided by Azure VM to safeguard your data and virtual machines. Azure Security Center and Azure Disk Encryption allow you to encrypt data both in transit and at rest. For identity and access control, you may also use firewalls, Microsoft Entra ID, and network security groups (NSGs).

· Cost-effectiveness: Pay-as-you-go pricing is available for Azure VM, letting you only pay for the resources you really need. By selecting the right virtual machine (VM) size, taking use of reserved instances for special pricing, and making use of Azure Hybrid Benefit to save money on Windows VMs, you may maximise expenses.

To concentrate on business goals without having to worry about infrastructure administration, Azure VM offers a dependable, scalable, and affordable option for hosting workloads and apps in the cloud.

[(Microsoft Azure, 2024c)](https://www.zotero.org/google-docs/?dgde1B)

Azure Application Gateway and Azure Front Door

Introduction:

Organisations are depending more and more on online solutions in today's digital environment to engage with consumers and drive corporate development. Nonetheless, there are several obstacles in the way of guaranteeing these online solutions like performance, scalability, and security. Azure provides Azure Application Gateway and Azure Front Door, two strong services that are intended to solve these issues and enable businesses to create scalable and secure online solutions.

Azure Application Gateway:

With features like web application firewall (WAF), SSL termination, and intelligent load balancing, Azure Application Gateway acts as a scalable and secure entry point for online applications. Web applications may be guaranteed high availability and excellent performance by using Application Gateway, which dynamically distributes traffic among several backend servers. Furthermore, by defending against frequent web-based attacks, its integrated WAF preserves the integrity of online apps and secures important data.

Azure Front Door:

Azure Front Door is a globally distributed, highly available content delivery network (CDN) that improves online application performance and security. Front Door's worldwide network of edge sites decreases latency and accelerates content delivery, resulting in a consistent user experience across regions. Furthermore, Front Door includes comprehensive security features such as DDoS protection and web application firewall (WAF) capabilities, which safeguard online applications from harmful attacks and vulnerabilities.

Solution Architecture:

Azure Application Gateway and Azure Front Door may be used to create online solutions with full security and scalability. Azure Front Door acts as a global entering point, improving traffic routing and speeding up content delivery. Azure Application Gateway serves as an additional layer of protection, providing advanced load balancing and implementing additional security measures such as SSL termination and WAF rules.

Key Benefits:

· Security: Azure Application Gateway and Azure Front Door provide comprehensive security capabilities including SSL termination, WAF, and DDoS protection, which protect online applications from attacks and vulnerabilities.

· Scalability: With built-in auto scaling features, both services can handle traffic volume changes and automatically scale resources to assure maximum efficiency.

· Performance: Azure Front Door's global network and content delivery optimizations reduce latency and accelerate content delivery, giving users a quick and responsive online experience.

· Simplicity: Organisations may reduce operational expenses and increase agility by using Azure managed services for deployment and administration.

Conclusion:

Azure Application Gateway and Azure Front Door are critical technologies for developing secure, scalable, and high-performance online applications in Azure. Organisations may efficiently solve the issues of modern web application delivery by integrating these services into solution designs, resulting in a seamless and secure online experience for users [(Microsoft Learn, 2024h, 2024i)](https://www.zotero.org/google-docs/?ASslby).

Azure Monitor For Comprehensive Cloud Monitoring

Introduction:

In today's dynamic cloud systems, good monitoring and resource management are critical to ensuring optimal performance, availability, and security. Azure Monitor is a strong cloud monitoring service that delivers information on the performance and health of Azure resources, apps, and workloads. This solution architecture explains how companies may use Azure Monitor to obtain complete monitoring and actionable insights into their Azure deployments.

Azure Monitor Overview:

Azure Monitor is a single monitoring solution that gives a complete picture of the Azure environment, that includes infrastructure, apps, and services. It provides a variety of monitoring features, including as metrics, logs, alarms, and dashboards, allowing companies to proactively monitor, diagnose, and fix issues.

Key Components:

· Metrics: Azure Monitor gathers and maintains performance information from Azure resources, allowing companies to track resource use, discover trends, and improve performance.

· Logs: Azure Monitor collects log data from Azure resources and apps to provide insights into operational activity, faults, and security occurrences. It can handle logs from a variety of sources, including Azure resources, apps, and custom logs.

· Alerts: Azure Monitor enables companies to set up alerts based on predefined or custom circumstances, allowing for proactive detection of errors and abnormalities.

· Dashboards: Azure Monitor provides configurable dashboards that provide companies an overview of monitoring data, allowing them to see critical metrics and trends.

Solution Architecture:

Azure Monitor is embedded into Azure's monitoring and management layer. It takes telemetry data from different Azure resources, apps, and services, combines and analyses it, and then provides actionable insights.

Key Benefits:

· Proactive Monitoring: Azure Monitor allows companies to proactively monitor the health, performance, and security of their Azure deployments, lowering the likelihood of downtime and service interruptions.

· Actionable Insights: Azure Monitor aggregates and analyses telemetry data to deliver meaningful insights into resource use, performance trends, and operational concerns, allowing companies to make well-informed choices and manage their Azure installations.

· Scalability: Azure Monitor adapts dynamically to meet the monitoring requirements of companies of all kinds, from small businesses to huge enterprises, assuring scalability and performance as Azure installations expand.

Conclusion:

Azure Monitor is a potent monitoring tool that gives businesses deep insight and access into their Azure deployments. Organisations may efficiently monitor, troubleshoot, and improve their Azure environments using Azure Monitor, guaranteeing peak performance, availability, and security [(Microsoft Learn, 2024j)](https://www.zotero.org/google-docs/?bfJryU).

Cloud Storage Solutions with Azure Storage

Introduction:

Organisations need scalable and dependable cloud storage solutions more than ever in the age of digital transformation in order to efficiently store, manage, and analyse data. Azure Storage is a feature-rich cloud storage solution that provides a variety of storage choices to accommodate the various demands of contemporary workloads and apps. This solution architecture describes how businesses may use Azure Storage to create cloud storage solutions that are dependable, scalable, and reasonably priced.

Azure Storage Overview:

Blobs, files, tables, queues, and other sorts of data may all be stored in scalable, secure, and highly accessible ways with Azure Storage, a fully managed cloud storage solution. Building adaptable and durable storage solutions is made possible for companies by its numerous storage tiers, strong security features, and smooth interaction with other Azure services.

Key Components:

· Blob Storage: A highly scalable object storage solution called Azure Blob Storage enables companies to store and handle unstructured data including logs, documents, photos, and videos.

· File Storage: With the help of fully managed file shares offered by Azure File Storage, companies can set up shared file systems that are accessible from a variety of virtual machines and apps.

· Table Storage: With Azure Table Storage, a NoSQL data store, companies can store structured data as tables and give apps scalable, low-latency access to the data.

· Queue Storage: With the scalable message queuing service offered by Azure Queue Storage, companies can create distributed apps with asynchronous communication patterns.

Solution Architecture:

The data storage layer of the program or solution incorporates Azure Storage. Supporting several data kinds and access patterns, it acts as the main data repository for organising and storing data.

Key Benefits:

· Scalability: With Azure Storage's almost limitless scalability, companies can adjust their storage capabilities on-the-fly to accommodate expanding data sets and shifting business requirements.

· Reliability: Because Azure Storage has high availability and redundancy built in, data is constantly accessible and safeguarded from datacenter outages and hardware malfunctions.

· Security: Role-based access control (RBAC), network security policies, and encryption at rest and in transit are just a few of the strong security features that Azure Storage provides to safeguard data from cyberattacks and unwanted access.

· Cost-effectiveness: Pay-as-you-go pricing and storage tiers are only two of the various pricing choices that Azure Storage offers, allowing companies to manage storage costs depending on data retention needs and usage patterns.

Conclusion:

Azure Storage is a cloud storage service that offers companies scalable, dependable, and reasonably priced storage options for a variety of workloads and data kinds. Organisations may create adaptable and durable storage solutions that satisfy the changing requirements of contemporary workloads and apps by utilising Azure Storage.[(Microsoft Learn, 2023i)](https://www.zotero.org/google-docs/?KMFt9g)

Identity Management With Microsoft Entra ID

Introduction:

In the current digital environment, protecting access to resources, apps, and data requires efficient identity management. Microsoft's cloud-based identity and access management solution, Microsoft Entra ID offers a full range of features to manage identities, impose security rules, and provide easy access to resources. This solution design shows how companies may use Microsoft Entra ID to create scalable and safe identity management systems.

Overview:

Microsoft Entra ID is a cloud-based identity and access management solution that offers features for on-premises and cloud resources in terms of permission, authentication, and identity protection. It is the foundation of Microsoft's cloud identity platform, which gives companies the ability to manage user identities, verify user identities, and restrict access to services and apps.

Key Components:

· Identity Management: For the purpose of managing user IDs, groups, and characteristics, Microsoft Entra ID offers a centralised identity store. It facilitates identity lifecycle management, self-service features, and user provisioning.

· Authentication: Password-based authentication, multi-factor authentication (MFA), and federation with external identity providers (IdPs) via industry-standard protocols like SAML, OAuth, and OpenID Connect are just a few of the authentication options that Microsoft Entra ID provides.

· Authorization: Organisations may designate access control policies and provide rights according to user roles and groups by using Microsoft Entra ID. In order to manage privileged access, it offers Privileged Identity Management (PIM), conditional access restrictions, and role-based access control (RBAC).

· Identity Protection: In order to assist businesses in identifying and reducing identity-related security risks, Microsoft Entra ID offers identity protection capabilities including risk-based conditional access, anomaly detection, and automated threat response.

· Integration: Single sign-on (SSO) and centralised access management are made possible by Microsoft Entra ID's seamless integration with Microsoft 365, Azure services, and hundreds of third-party apps and services via Microsoft Entra ID Connect and Microsoft Entra ID Application Proxy.

Azure Cache for Redis:

Azure Cache for Redis is a high-performing caching service provided by Microsoft Azure, designed to deliver 99.9% availability. This service offers in-memory data storage that can significantly improve the performance and scalability of the applications by reducing latency and increasing throughput. Azure Cache for Redis supports data replication, persistence, and clustering, ensuring data reliability and fault tolerance. By leveraging Redis, an open-source, in-memory data structure store, Azure Cache for Redis enables developers to build responsive and highly available applications that can handle large workloads effectively. With its robust features and high availability guarantee, Azure Cache for Redis is a valuable tool for optimising application performance in the cloud.

[(Microsoft Azure, 2023; Microsoft Learn, 2023k)](https://www.zotero.org/google-docs/?yKVsQb)

Solution Architecture:

Microsoft Entra ID is the main identity and access management component in a typical solution architecture, offering cloud and hybrid environments authentication, authorization, and identity protection features.

Key Benefits:

· Security: To defend against identity-related security risks, Microsoft Entra ID offers strong security features including conditional access, MFA, and identity protection.

· Scalability: As businesses expand, Microsoft Entra ID's dynamic scaling ensures performance and scalability for millions of users and devices.

· Productivity: Microsoft Entra ID reduces IT costs and increases user productivity by enabling seamless access to apps and services through SSO.

· Compliance: Microsoft Entra ID's ability to audit user activity, enforce access control restrictions, and identify security concerns enables businesses to comply with regulatory obligations.

Conclusion:

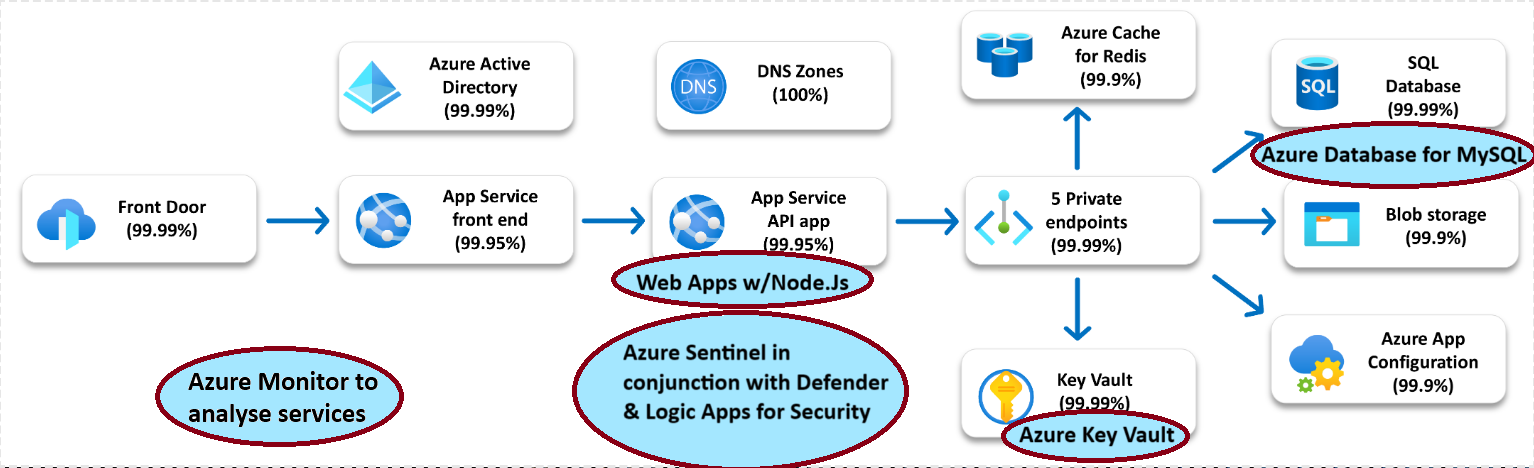
Secure, scalable, and compliant identity management solutions are offered to companies by Microsoft Entra ID, a potent identity and access management service. Organisations may achieve seamless access to resources, apps, and services, enforce security standards, and centralise identity management by utilising Microsoft Entra ID[(Microsoft Learn, 2023j)](https://www.zotero.org/google-docs/?FJvUKa).

Proposed Solutions Architecture For Application

With the help of many Azure services, the suggested solutions architecture supports an application ecosystem with a comprehensive infrastructure. Among the essential elements are **Azure Database for MySQL** for storing data, a Node.js **Web App** operating on Azure virtual machines, **Azure Key Vaults** for safe key administration, **Azure Application Gateway** and **Azure Front Door** for traffic control and load balancing, **Azure Storage** for storing files and objects, **Microsoft Entra ID** for identity management, and **Azure Monitor** for monitoring and logging. For continuous integration, delivery, and deployment, **DevOps** techniques are included into the whole system. Along with enabling effective administration and monitoring of the entire system, this design guarantees scalability, stability, security, and performance for the application. The suggested architecture may be easily expanded to include **Azure Cache for Redis**, improving scalability and speed.

Responses times and overall application performance can be enchanted by a large reduction in database load and latency by doing so.

By using **Azure Sentinel** in conjunction with **Microsoft Defender for Cloud** we can increase the security of our VM and its services. By utilising the secure score aspects we can quickly ensure that our application is configured correctly and securely in a rapid manner and from there be able to monitor any security concerns with Sentinel. By using **Logic Apps** automation services, any security concerns and be automatically address to prevent further exploits



# (Image showcasing that the services we have suggest follows microsoft's recommendation for a reliable web application)

Breakdown Of Tasks

Oluwamayowa Akinola:

1. A03:2021-Injection:
2. A04:2021 – Insecure Design
3. A07:2021 – Identification and Authentication Failures
4. Proposed Solutions Architecture (Part 2)
5. SQL databases
6. Web App > Nodejs server
7. Microsoft defender for cloud (formerly known as - Azure security center)
8. Microsoft sentinel
9. Disk Encryption Options
10. Zero Trust framework
11. Microsoft Best practices
12. Breakdown of Tasks
13. Formatting the document
14. Inserting reference to Havard standard

Alvis Auzins:

1. A05:2021-Security Misconfiguration
2. A06:2021- Vulnerable and Outdated Components
3. A08:2021 – Software and Data Integrity Failures
4. Azure VM
5. Azure Application Gateway and Azure Front Door
6. Azure Monitor for Comprehensive Cloud Monitoring
7. Cloud Storage Solutions with Azure Storage
8. Identity Management with Microsoft Entra ID

Samuel Dauda:

1. A02:2021-Cryptographic failures:
2. A10:2021 – Server-Side Request Forgery (SSRF)
3. SSh keys - Azure key vaults
4. Static web apps
5. Dev Ops
6. Azure functions
7. Logic Apps
8. Azure Cache for Redis

Team (lead person to be highlighted) :

* A01:2021 – Broken Access Control (Mayowa)
* A09:2021 – Security Logging and Monitoring Failures (Alvis)
* Reference for tools and applications (Alvis & Mayowa)
* Proposed Solutions Architecture For Application (Alvis & Mayowa)

Reference For Tools And Applications

* Juice Shop - Kimminich, B., OWASP, n.d. OWASP Juice Shop | OWASP Foundation [WWW Document]. URL<https://owasp.org/www-project-juice-shop/> (accessed 2.13.24).
* Snyk - Snyk, n.d. Snyk | Developer security | Develop fast. Stay secure. [WWW Document]. URL<https://snyk.io/> (accessed 3.21.24).
* Zap - Zap Dev Team, 2024. The ZAP Homepage [WWW Document]. URL<https://www.zaproxy.org/> (accessed 3.21.24).
* Zotero - Zotero, n.d. Zotero | Your personal research assistant [WWW Document]. URL<https://www.zotero.org/> (accessed 3.21.24).

References:

AdRoll, 2024. Email Verification: Verifying your email address when you sign up [WWW Document]. AdRoll Help Center. URL<https://help.adroll.com/hc/en-us/articles/360057993352-Email-Verification-Verifying-your-email-address-when-you-sign-up> (accessed 3.21.24).

Albeniz, Z., Morgenroth, S., Yildirimkaya, U., 2023. HTTP Security Headers and How They Work [WWW Document]. Invicti. URL<https://www.invicti.com/white-papers/whitepaper-http-security-headers/> (accessed 3.21.24).

Apostrophe Technologies, 2024. sanitize-html [WWW Document]. URL<https://www.npmjs.com/package/sanitize-html> (accessed 3.21.24).

auth0, 2024. node-jsonwebtoken [WWW Document]. URL<https://github.com/auth0/node-jsonwebtoken> (accessed 3.21.24).

auth0, 2023. express-jwt [WWW Document]. npm. URL<https://www.npmjs.com/package/express-jwt> (accessed 3.21.24).

Brakefield, J., 2023. Private Open AI Service in Azure with Web UI Front End! Medium. URL<https://jbrake30.medium.com/private-open-ai-service-in-azure-with-web-4ce0b70dd76c> (accessed 3.21.24).

Brix, 2024. brix/crypto-js [WWW Document]. URL<https://github.com/brix/crypto-js> (accessed 3.21.24).

c58, Artemyev, A., 2016. marsdb [WWW Document]. URL<https://www.npmjs.com/package/marsdb> (accessed 3.21.24).

Caniuese.com, 2024. Content Security Policy 1.0 | Can I use... Support tables for HTML5, CSS3, etc [WWW Document]. URL<https://caniuse.com/contentsecuritypolicy> (accessed 3.21.24).

Cloudflare, n.d. Zero Trust security | What is a Zero Trust network? [WWW Document]. Zero Trust security. URL<https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/> (accessed 3.21.24).

Common Weakness Enumeration, 2024a. CWE - CWE-548: Exposure of Information Through Directory Listing (4.14) [WWW Document]. URL<https://cwe.mitre.org/data/definitions/548.html> (accessed 3.21.24).

Common Weakness Enumeration, 2024b. CWE - CWE-829: Inclusion of Functionality from Untrusted Control Sphere (4.14) [WWW Document]. URL<https://cwe.mitre.org/data/definitions/829.html> (accessed 3.21.24).

Content Security Policy (CSP), 2023. Content-Security-Policy (CSP) Header Quick Reference [WWW Document]. URL<https://content-security-policy.com/> (accessed 3.21.24).

Cuff, S., 2021a. Azure Security product name changes – Microsoft Ignite November 2021 [WWW Document]. TECHCOMMUNITY.MICROSOFT.COM. URL<https://techcommunity.microsoft.com/t5/itops-talk-blog/azure-security-product-name-changes-microsoft-ignite-november/ba-p/3004418> (accessed 3.21.24).

Cuff, S., 2021b. What’s the difference between Azure Security Center, Azure Defender and Azure Sentinel? [WWW Document]. TECHCOMMUNITY.MICROSOFT.COM. URL<https://techcommunity.microsoft.com/t5/itops-talk-blog/what-s-the-difference-between-azure-security-center-azure/ba-p/2155188> (accessed 3.21.24).

Edwards, A., 2021. Comparing Azure Static Web Apps vs Azure WebApps vs Azure Blob Storage Static Sites [WWW Document]. Azure DevOps Blog. URL<https://devblogs.microsoft.com/devops/comparing-azure-static-web-apps-vs-azure-webapps-vs-azure-blob-storage-static-sites/> (accessed 3.21.24).

Hike, 2023. What to Do If My Account Is Locked [WWW Document]. Hike. URL<https://help.hikeup.com/portal/en/kb/articles/what-to-do-if-my-account-is-locked> (accessed 3.21.24).

IBM, 2024a. What is SIEM? | IBM [WWW Document]. IBM - SIEM. URL<https://www.ibm.com/topics/siem> (accessed 3.21.24).

IBM, 2024b. What is SOAR (security orchestration, automation and response)? | IBM [WWW Document]. IBM - SOAR. URL<https://www.ibm.com/topics/security-orchestration-automation-response> (accessed 3.21.24).

IBM, 2021. IBM Documentation - Types of Injection attacks [WWW Document]. Types of Injection attacks. URL<https://www.ibm.com/docs/en/snips/4.6.0?topic=categories-injection-attacks> (accessed 3.21.24).

Johnson, W., auth0, Okta, 2022. RS256 vs HS256 What’s the difference? [WWW Document]. Auth0 - Blog. URL<https://auth0.com/blog/rs256-vs-hs256-whats-the-difference/> (accessed 3.21.24).

Kimminich, B., OWASP, 2024. OWASP Juice Shop | OWASP Foundation [WWW Document]. URL<https://owasp.org/www-project-juice-shop/> (accessed 2.13.24).

Lee, K., Sjöberg, S., Narayanan, A., 2022. Password policies of most top websites fail to follow best practices.

M, S., 2023. Deploy Azure Disk Encryption for Data Disk. Azure365Pro.com. URL<https://www.azure365pro.com/deploy-azure-disk-encryption-for-data-disk/> (accessed 3.21.24).

Medley, J., West, M., 2024. Content security policy | Articles [WWW Document]. web.dev. URL<https://web.dev/articles/csp> (accessed 3.21.24).

Microsoft Azure, 2024a. App Service — Build and Host Web Apps | Microsoft Azure [WWW Document]. URL<https://azure.microsoft.com/en-us/products/app-service> (accessed 3.21.24).

Microsoft Azure, 2024b. Azure documentation [WWW Document]. URL<https://learn.microsoft.com/en-us/azure/> (accessed 3.21.24).

Microsoft Azure, 2024c. Logic App Service – IPaaS | Microsoft Azure [WWW Document]. URL<https://azure.microsoft.com/en-us/products/logic-apps> (accessed 3.21.24).

Microsoft Azure, 2023. Azure Cache for Redis | Microsoft Azure [WWW Document]. URL<https://azure.microsoft.com/en-us/products/cache> (accessed 3.21.24).

Microsoft Azure, 2019. Microsoft Azure Marketplace [WWW Document]. Azure Security Center. URL<https://azuremarketplace.microsoft.com/en-us/marketplace/apps/microsoft.azuresecuritycenter?tab=overview> (accessed 3.21.24).

Microsoft Learn, 2024a. Azure Application Gateway documentation [WWW Document]. URL<https://learn.microsoft.com/en-us/azure/application-gateway/> (accessed 3.21.24).

Microsoft Learn, 2024b. Azure Front Door and CDN Documentation [WWW Document]. URL<https://learn.microsoft.com/en-us/azure/frontdoor/> (accessed 3.21.24).

Microsoft Learn, 2024c. Azure Monitor documentation - Azure Monitor [WWW Document]. URL<https://learn.microsoft.com/en-us/azure/azure-monitor/> (accessed 3.21.24).

Microsoft Learn, 2024d. Azure SQL documentation - Azure SQL [WWW Document]. Azure SQL documentation. URL<https://learn.microsoft.com/en-us/azure/azure-sql/?view=azuresql> (accessed 3.21.24).

Microsoft Learn, 2024e. Create a single database - Azure SQL Database [WWW Document]. Create a single database - Azure SQL Database. URL<https://learn.microsoft.com/en-us/azure/azure-sql/database/single-database-create-quickstart?view=azuresql> (accessed 3.21.24).

Microsoft Learn, 2024f. Enable Azure Disk Encryption for Windows VMs - Azure Virtual Machines [WWW Document]. Enable Azure Disk Encryption for Windows VMs - Azure Virtual Machines. URL<https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-overview> (accessed 3.21.24).

Microsoft Learn, 2024g. Microsoft Defender for Cloud documentation - Microsoft Defender for Cloud [WWW Document]. Microsoft Defender for Cloud documentation. URL<https://learn.microsoft.com/en-us/azure/defender-for-cloud/> (accessed 3.21.24).

Microsoft Learn, 2024h. Overview of managed disk encryption options - Azure Virtual Machines [WWW Document]. Overview of managed disk encryption options - Azure Virtual Machines. URL<https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview> (accessed 3.21.24).

Microsoft Learn, 2024i. Review cloud security posture in Microsoft Defender for Cloud - Microsoft Defender for Cloud [WWW Document]. Review cloud security posture in Microsoft Defender for Cloud - Microsoft Defender for Cloud. URL<https://learn.microsoft.com/en-us/azure/defender-for-cloud/overview-page> (accessed 3.21.24).

Microsoft Learn, 2024j. Server-side encryption of Azure managed disks - Azure Virtual Machines [WWW Document]. Server-side encryption of Azure managed disks - Azure Virtual Machines. URL<https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption> (accessed 3.21.24).

Microsoft Learn, 2023a. Azure Database for MySQL - Flexible Server documentation [WWW Document]. Azure Database for MySQL - Flexible Server documentation. URL<https://learn.microsoft.com/en-us/azure/mysql/> (accessed 3.21.24).

Microsoft Learn, 2023b. Azure portal - Enable customer-managed keys with SSE - managed disks - Azure Virtual Machines [WWW Document]. Azure portal - Enable customer-managed keys with SSE - managed disks - Azure Virtual Machines. URL<https://learn.microsoft.com/en-us/azure/virtual-machines/disks-enable-customer-managed-keys-portal> (accessed 3.21.24).

Microsoft Learn, 2023c. Basic web application - Azure Reference Architectures [WWW Document]. Basic web application - Azure Reference Architectures. URL<https://learn.microsoft.com/en-us/azure/architecture/web-apps/app-service/architectures/basic-web-app> (accessed 3.21.24).

Microsoft Learn, 2023d. Best practices for protecting your organization - Microsoft Defender for Cloud Apps [WWW Document]. Best practices for protecting your organization - Microsoft Defender for Cloud Apps. URL<https://learn.microsoft.com/en-us/defender-cloud-apps/best-practices> (accessed 3.21.24).

Microsoft Learn, 2023e. Introduction to Azure Storage - Cloud storage on Azure [WWW Document]. URL<https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction> (accessed 3.21.24).

Microsoft Learn, 2023f. Overview - Azure App Service [WWW Document]. URL<https://learn.microsoft.com/en-us/azure/app-service/overview> (accessed 3.21.24).

Microsoft Learn, 2023g. Reliable web app pattern for .NET - Plan the implementation - Azure Reference Architectures [WWW Document]. Reliable web app pattern for .NET - Plan the implementation - Azure Reference Architectures. URL<https://learn.microsoft.com/en-us/azure/architecture/web-apps/guides/reliable-web-app/dotnet/plan-implementation> (accessed 3.21.24).

Microsoft Learn, 2023h. Security best practices and patterns - Microsoft Azure [WWW Document]. URL<https://learn.microsoft.com/en-us/azure/security/fundamentals/best-practices-and-patterns> (accessed 3.21.24).

Microsoft Learn, 2023i. What is Azure Cache for Redis? - Azure Cache for Redis [WWW Document]. What is Azure Cache for Redis? - Azure Cache for Redis. URL<https://learn.microsoft.com/en-us/azure/azure-cache-for-redis/cache-overview> (accessed 3.21.24).

Microsoft Learn, 2023j. What is Microsoft Entra ID? - Microsoft Entra [WWW Document]. URL<https://learn.microsoft.com/en-us/entra/fundamentals/whatis> (accessed 3.21.24).

Microsoft Learn, 2023k. What is Microsoft Sentinel? [WWW Document]. What is Microsoft Sentinel? URL<https://learn.microsoft.com/en-us/azure/sentinel/overview> (accessed 3.21.24).

Microsoft Learn, 2016. Reducing MIME type security risks (Windows) [WWW Document]. URL<https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)> (accessed 3.21.24).

Microsoft Security, 2024. Zero Trust Model - Modern Security Architecture | Microsoft Security [WWW Document]. Zero Trust Model. URL<https://www.microsoft.com/en-ie/security/business/zero-trust> (accessed 3.21.24).

Microsoft Security, n.d. What Is Two-Factor Authentication (2FA)? | Microsoft Security [WWW Document]. URL<https://www.microsoft.com/en-ie/security/business/security-101/what-is-two-factor-authentication-2fa> (accessed 3.21.24).

Mozilla Developer, 2024. Content Security Policy (CSP) - HTTP | MDN [WWW Document]. URL<https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP> (accessed 3.21.24).

Mozilla Developer, 2023. X-Frame-Options - HTTP | MDN [WWW Document]. URL<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options> (accessed 3.21.24).

Okta, 2024. What Is Token-Based Authentication? | Okta [WWW Document]. URL<https://www.okta.com/identity-101/what-is-token-based-authentication/> (accessed 3.21.24).

Opera Add-Ons, 2023. Buster: Captcha Solver for Humans [WWW Document]. Opera add-ons. URL<https://addons.opera.com/extensions/details/buster-captcha-solver-for-humans/> (accessed 3.21.24).

OWASP, 2024a. OWASP Developer Guide | Principles of Security | OWASP Foundation [WWW Document]. URL<https://owasp.org/www-project-developer-guide/draft/foundations/security_principles/> (accessed 3.21.24).

OWASP, 2024b. Secure Product Design - OWASP Cheat Sheet Series [WWW Document]. OWASP Cheat Sheet Series. URL<https://cheatsheetseries.owasp.org/cheatsheets/Secure_Product_Design_Cheat_Sheet.html> (accessed 3.21.24).

Owasp, 2024. Security Headers | OWASP Foundation [WWW Document]. URL<https://owasp.org/www-community/Security_Headers> (accessed 3.21.24).

OWASP, 2021a. A01 Broken Access Control - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A01_2021-Broken_Access_Control/> (accessed 3.21.24).

OWASP, 2021b. A02 Cryptographic Failures - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A02_2021-Cryptographic_Failures/> (accessed 3.21.24).

OWASP, 2021c. A03 Injection - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A03_2021-Injection/> (accessed 3.21.24).

OWASP, 2021d. A04 Insecure Design - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A04_2021-Insecure_Design/> (accessed 3.21.24).

OWASP, 2021e. A05 Security Misconfiguration - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A05_2021-Security_Misconfiguration/> (accessed 3.21.24).

OWASP, 2021f. A06 Vulnerable and Outdated Components - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/> (accessed 3.21.24).

OWASP, 2021g. A07 Identification and Authentication Failures - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/> (accessed 3.21.24).

OWASP, 2021h. A08 Software and Data Integrity Failures - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/> (accessed 3.21.24).

OWASP, 2021i. A09 Security Logging and Monitoring Failures - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/> (accessed 3.21.24).

OWASP, 2021j. A10 Server Side Request Forgery (SSRF) - OWASP Top 10:2021 [WWW Document]. OWASP. URL<https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/> (accessed 3.21.24).

Owasp Cheat Sheet Series, 2024. Content Security Policy - OWASP Cheat Sheet Series [WWW Document]. URL<https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html> (accessed 3.21.24).

Prins, C.W., 2018. Adjusting the Password Policy [WWW Document]. 4me. URL<https://www.4me.com/blog/password-policy/> (accessed 3.21.24).

Snyk, 2024. Snyk | Developer security | Develop fast. Stay secure. [WWW Document]. URL<https://snyk.io/> (accessed 3.21.24).

Snyk Learn, 2024. What is code injection? | Tutorial & examples [WWW Document]. Snyk Learn. URL<https://learn.snyk.io/lesson/malicious-code-injection/> (accessed 3.21.24).

SolarWinds, n.d. SQL Injection Monitor - Detecting SQL Injection Attacks | SolarWinds [WWW Document]. URL<https://www.solarwinds.com/security-event-manager/use-cases/sql-injection-attack> (accessed 3.21.24).

StackShare, 2023. Azure Database for MySQL vs Azure SQL Database | What are the differences? [WWW Document]. StackShare. URL<https://stackshare.io/stackups/azure-database-for-mysql-vs-azure-sql-database> (accessed 3.21.24).

w3.org, West, M., Sartori, A., 2024. Content Security Policy Level 3.

Warner, T., 2023. Differentiate Microsoft Defender for Cloud and Microsoft Sentinel [WWW Document]. Tekkigurus. URL<https://www.tekkigurus.com/differentiate-microsoft-defender-for-cloud-and-microsoft-sentinel/> (accessed 3.21.24).

Zap Dev Team, 2024. The ZAP Homepage [WWW Document]. URL<https://www.zaproxy.org/> (accessed 3.21.24).

Zotero, 2024. Zotero | Your personal research assistant [WWW Document]. URL<https://www.zotero.org/> (accessed 3.21.24).