A Best Practices Guide to Project Security...

workingmouse

workingmouse

Version → 1.0

01 APRIL
2022

Contents

Introduction **4**

Application Security **5**

Standards and Practices 6

Authentication, Authorisation and Auditing/Accounting 11

Application Architecture 12

User Management Security 14

Operational Security **16**

Deployment Considerations 17

Cloud 18

Common Migrations 25

Access Control 26

Endpoint Security **28**

Vulnerabilities 29

Data Integrity 29

Managing Endpoint Security 29

Glossary **30**

Who Are We? **32**

Introduction

Web security has been continuously improving over the last decade. The security options available have allowed industries such as banking and defence to move many of its tools to online applications. This has brought these industries a variety of advantages that come with using online applications over desktop applications. Despite this, there are several security risks that are important to identify and mitigate.

The purpose of this document is to address concerns regarding the security for web applications which contain sensitive or restricted data.

By summarising policies, processes and tools used throughout the software development cycle, and highlighting common risks, this guide will provide recommendations for technical project leaders who want to ensure they have considered the most common security risks and the best methods of addressing them.

While it is difficult to protect from infinite possibilities of security attacks, we believe we have a refined process in place that enables us to build an application in a highly secure cloud-based web environment.

Application Security

Application security refers to an applications' ability to defend against attacks independent of the environment to which it is deployed. When discussing application security, we make assumptions regarding the security of other external factors such as the application's hosting to simplify the discussion.

The other types of factors are addressed in the Operational Security section.

Key Assumptions:

- A malicious actor does not have access to the environment in which the application is running
- A malicious actor's only ingress is via the public domain and port, i.e. only on https://applicationdomain.com

Standards and Practices

We recommend following several processes to ensure your organisation aligns with the industry's best practices.

Application Security Verification Standard

The Application Security Verification Standard (ASVS) is a project developed by OWASP that aims to provide an open application security standard for web apps and web services of all types.

The Application Security Verification Standard defines three security verification levels, with each level increasing in depth.

- ASVS Level 1 is for low assurance levels and is completely penetration testable using humans.
- ASVS Level 2 is for applications that contain sensitive data, which requires protection and is the recommended level for most applications.
- ASVS Level 3 is for the most critical applications - applications that perform high-value transactions, contain sensitive medical data, or any application that requires the highest level of trust.

Level 1 can be mostly integration tested using selenium or other frameworks. Even if Level 1 allows black box testing to occur, it is not an effective assurance activity.

Level 2 and 3 require access to documentation, source code, configuration, and the people involved in the development process.

WorkingMouse utilises this standard heavily as our key source of recommendations for building highly secure software. We integrate the recommendations in our development lifecycle from scoping, delivery, through to deployment and support.

While many of the recommendations are relevant to application security, many are operational in nature and as such will be addressed in the Operational Security section. The recommendations,

while focused on web applications, are often applicable to software development of all types.

Security Scanning During Development

One of the most important attributes of security testing is coverage. To assess the security of an application, an automated scanner must be able to accurately interpret that application. Automated vulnerability scanning allows developers to always be on the lookout for new attack paths that attackers can use to access a web application or the data behind it. We can also integrate security testing tools in our build pipeline.

There are two types of security testing:

- **Dynamic application security test** (DAST, also known as black box testing) is a type of testing that looks for security vulnerabilities by safely exploiting a running application from the outside. This type of testing is not dependent on the framework or programming language used.
- **Static application security test** (SAST, also known as white box testing) is a type of testing that includes code analysers. It tests the source code for vulnerabilities by identifying the common patterns in it. These tools are language-specific and can be only used by the development team.

Unfortunately, automated tools and online scans cannot completely test against ASVS criteria without human assistance. An automated security testing tool is encouraged to provide as much coverage as possible. A large majority of ASVS requirements in Level 1 can be performed using automated tests. However, many requirements in Level 2 and Level 3 requirements are not amenable to automated tests due to the sensitivity of the data.

As a standard, WorkingMouse recommends basic automated SAST testing against applications.

Further Reading

- Static Application Security Testing

Application Third-Party Dependency Auditing and Patching

A key part of creating a highly secure application is the monitoring and patching of vulnerabilities as they become known.

> *Dependency management is critical to the safe operation of any application of any type. Failure to keep up to date with outdated or insecure dependencies is the root cause of the largest and most expensive attacks to date.*
>
> *- OWASP ASVS*

WorkingMouse recommends using autom`ation tools to identify dependencies used within the application that have outstanding publicly disclosed vulnerabilities so that the application can quickly be patched. Wherever possible, the tools used are those maintained by the organisations that develop the technology to maximise their reliability.

For example, the .NET command line tool (CLI) package command provides a mechanism for auditing our server-side dependencies. As this is developed and maintained by Microsoft, who also develops and maintains the C# language, a certain degree of reliability can be expected. For our client-side package, the built-in yarn audit tool is used for the same reasons with the same benefits. Both examples operate online and, as such, maintain an up-to-date list of exposed vulnerabilities to check against.

This continual maintenance is a requirement to ensure that an application stays secure, even after the initial development.

Codebots

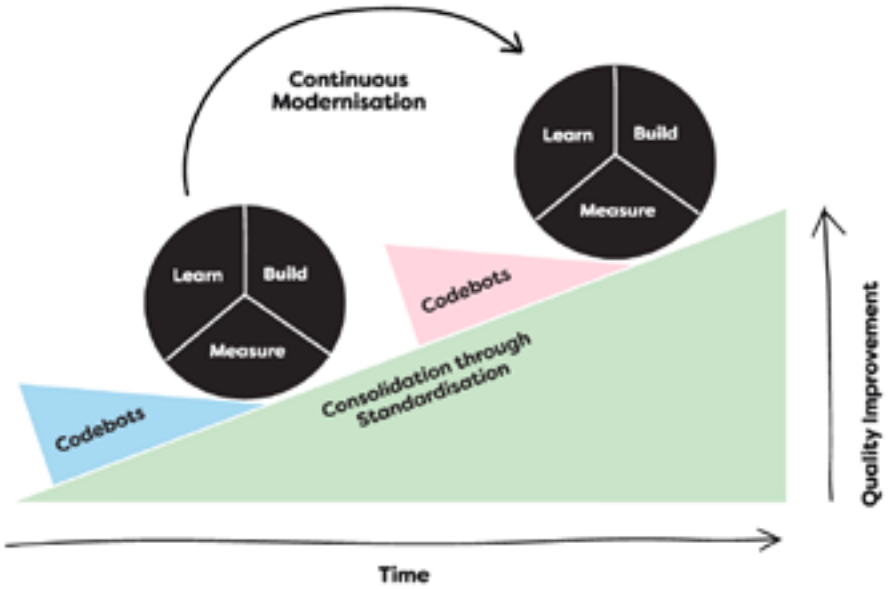
WorkingMouse makes use of the Codebots' App Studio and AI Lab development suite to develop high quality and secure applications.

A custom bot (C#Bot) is used to set the standard of code quality and functions in applications. The application's information architecture is then modelled out in App Studio, that's powered by C#Bot to write the source code.

Building with Codebots includes benefits such as:

- Powerful user management and security framework
- Constant security updates to address vulnerabilities discovered through penetration testing/vulnerability disclosures.

As C#Bot is used by many applications at WorkingMouse, vulnerabilities discovered by security audits and penetration testing performed for these applications result in updates back to the C#Bot itself to ensure all applications benefit.



Please see the User Management and Security section for more details on the user management and security framework.

Security Assessment and Penetration Testing

Before releasing the application into a production environment, an added security measure is to validate the quality of the application security thoroughly by hiring a penetration testing service. A penetration testing service (or pen test) is a form of ethical cybersecurity assessment designed to identify, through safe exploitation, vulnerabilities affecting computer networks, systems, applications, and websites so that any weaknesses discovered can be addressed to mitigate the risk of a malicious attack. Ultimately, the aim is to help shape cyber strategies and frameworks: penetration testing helps test, validate or invalidate the efficiency of defensive controls and determine what needs to be done to bolster them. If any vulnerabilities are discovered, we could address these issues through new tickets before releasing to the production version of the application.

WorkingMouse highly recommends the process of hiring independent penetration testing services. They can perform a suit of SAST and DAST tests, both looking for known vulnerabilities, and doing some white-hat hacking to see if they can break into the application. This will give an independent third-party opinion on the application security.

Multi-factor Authentication

According to the ASVS V2 Authentication section, applications should strongly encourage users to enrol in multi-factor authentication. By having multi-factor authentication, it can tick off multiple password and security criteria in the ASVS list.

C#Bot already has 2-factor authentication built-in on the server-side. If selected, this can be implemented in an application and allows users to configure and use a second factor when authenticating.

Authentication, Authorisation and Auditing/Accounting

AAA security stands for authentication, authorisation, and auditing.

Authentication is the first step in the AAA security process and describes the network or applications way of identifying a user and ensuring the user is whom they claim to be. The user enters a valid username and password before they are granted access; each user must have a unique set of identification information. Identification can be established via passwords, single sign-on (SSO) systems, biometrics, digital certificates, and public key infrastructure.

Authorisation refers to the process of enforcing policies, such as determining the qualities of activities, resources, or services a user is permitted to use. Authorisation usually occurs within the context of authentication; once you have been authenticated, AAA security authorisation assembles the set of attributes that describe what you are authorised to perform.

Auditing/Accounting measures the resources users consume during access to a network or application, logging session statistics and user information including session duration, and data sent and received. Usage information is used for authorisation control, billing, trend analysis, resource utilisation, and capacity planning activities.

AAA security enables mobile and dynamic security. Without AAA security, a network must be statically configured in order to control access. IP addresses must be fixed, systems cannot move, and connectivity options must be well defined. The proliferation of mobile devices and the diverse network of consumers with their varied network access methods generates a great demand for AAA security.

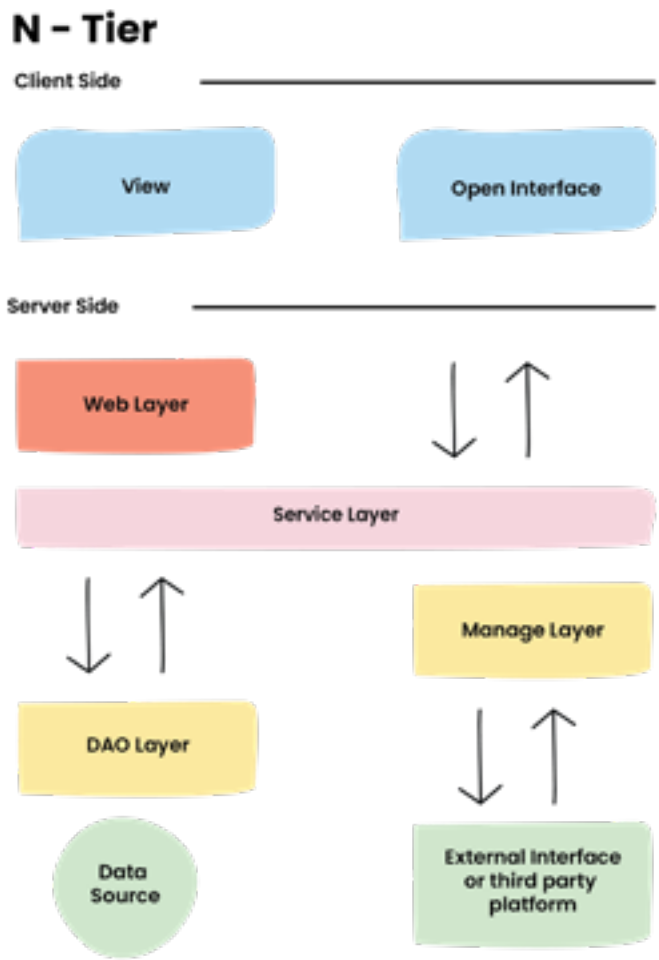
Application Architecture

When developing an application to be secure, it is important to consider the architecture.

To ensure that the architecture of each of our applications follows best practices and is consistent, WorkingMouse recommends writing applications into an N-tier architecture, which is a powerful general purpose application architecture that has strengths in clarity and separation of concerns. The Codebots development platform helps build applications to this architecture.

The key benefit of the N-tier architecture within the context of security is the ability to define clear bounded contexts within the application that can be secured independently from each other.

Within the context of a project, there are only two points of ingress within the n-tier structure. The web-layer and the external interface/third-party platform.



The Web Layer

Within a project, the web layer is represented by a set of APIs that allow the view (the client-side) to transfer data between the data source (database) and the user. These API endpoints provide a key vector for attack from a malicious actor. To protect against this, the bounded context of the service layer allows for the introduction of middleware to shield the internal workings of the application. For details on how this middleware works, please refer to 'User management and security'.

Additionally, the API endpoints allow for strict control over what data is exposed and prevent the internal workings of the application from becoming available to attackers.

In the worst case, an attacker can only gain insight about the application based on the responses to requests made through the API, ensuring any server-side logic/ code remains a black box to malicious third-party actors attempting to steal IP.

To gain access to IP protected by the application, an attacker would need to gain remote access to the application server. Mitigations and protections for this vector of attack can be noted in the Operational security section.

External Interface Layer

It is common in many applications to integrate with a third-party or external interface.

These integrations provide another avenue for attack.

There are two common methods of integration:

1. Integration from Third-Party to Application -> API requests from the third-party application to the application
2. Integration from the Application to the Third-Party -> Typically API requests from the application to the third-party

In the first method, the application becomes a passive entity that serves requests over the web layer. This is the most risk prone as it allows the integrating party to control the request structure and format, and it relies on the protections built into the web layer to ensure that only the requested/allowed information can be transmitted.

In the second method, the application becomes the controlling entity and as a result is less risky.

In both scenarios, WorkingMouse recommends to follow a minimum trust model. That is, we only trust the third-party as much as is strictly required, to ensure the successful functioning of the integration. It is recommended to follow this model to mitigate the risk that the third-party may become compromised and as such, any access allowed by the application to the third-party could also be gained by an attacker.

User Management and Security

WorkingMouse uses the user management security framework and configuration tools provided by the Codebots suite to maximise the protection against malicious actors.

Security Diagram

The security diagram is a configuration tool in Codebots that provides the AAA of security out of the box. By default, no resource can be accessed without being authenticated, and all operations are audited. Using the security diagram, a developer can define access to pages and business entities within the application. It allows for static security rules to be defined before they are written out in code.

The benefits of static, bot-written security rules are as follows:

- Bot-written ACLs (access-control list) are heavily tested across many versions and projects
- Security rules are documented in the security diagram (tight coupling ensures that the documentation reflects the implemented code)
- Clear and concise access controls

Security framework

C#Bot applications come out of the box with a full stack security framework that provides a combination of strict access control and user experience controls. The security controls are model based (from the Codebots security diagram) with ACL's being defined on a per-entity basis. While ACL's are core to the security framework, they are not the limit of what is possible.

The client-side of an application is not a reliable place to verify security, this is instead the duty of the server-side which includes user authentication, CRUD permissions and other considerations. However, it is important for the client-side to reflect the security rules.

Custom security

On top of security rules implemented by the codebot, based on the features required, developers implement custom ACL filtering security code. These requirements are discovered and documented during scope.

Operational Security

Operational security refers to the security of the deployed environment. It is involved with ensuring that malicious actors cannot and do not gain access outside the prescribed means dictated by the application.

Key Assumptions:

- Access via the prescribed ingress points (as dictated by the application) is correct and secure.

Deployment Considerations

An application can be deployed using a variety of different methods. It is extremely difficult for an application to be completely secure despite any inherit insecurity of its execution environment. As such it is often necessary for an application to place a degree of trust in the environment in which it has been deployed. To ensure that a system remains secure, both the application and execution environment need to be made secure in conjunction and independently.

Trusted Execution Environment

Applications require computer hardware to operate. This hardware includes CPU registers, memory, IO and storage. When an application executes, it relies on this hardware to reliably and securely hold or transmit data that is used during operation. This data can be privileged and highly sensitive.

Access to this hardware, physical or remote, can allow an attacker to compromise the application. Therefore, strict controls need to be put in place to ensure that an attacker cannot gain access to this environment.

Security is based upon a chain of trust. For an environment to be considered trustworthy, we must trust whoever has control of it. This trust is twofold - we must trust that the correct mitigations have been put into place to ensure the security, and we must trust that the party in question is not malicious. In the case of cloud environments, this typically results in a combination of trusting Microsoft, Amazon or Google and the managing party which can be WorkingMouse or a third-party. In the case of an on-premises server or workstation, we must trust whoever has access to it. This is usually our end users.

If it is difficult to trust these parties. Some of the possible mitigations that can be put into place to ensure that an environment is highly secure are as follows:

- Physical servers or workstations - Physical barriers such as locked rooms and air-gapping machines (ensuring they are not connected to any networks or the Internet), plus all the mitigations that apply to remote servers.

- Remote servers, like those used by the cloud - Utilisation of concepts such as VPC's (virtual private clouds), VPN's (virtual private networks) and locking down of certain network routes. We can typically trust the security in place on the physical servers.

All the mentioned mitigations are common practice and should be considered when discussing environment security.

Cloud

Cloud deployment involves the utilisation of a cloud service provider such as Azure (by Microsoft) or AWS (by Amazon). This document references Azure as an example but can be applied to any cloud platform.

When deploying to the cloud, we combine a trust for the security models and mitigations put in place by the provider (security processes in their data centres and resource isolations between shared services i.e. physical hardware) with due diligence processes and mitigations put in place by the managing party.

Of the deployment options, the cloud section will be the most compressive due to it being the most common chosen path for WorkingMouse customers. As such, WorkingMouse has an extensive set of processes and policies regarding ensuring that cloud environments remain secure.

Critical Deployment Components

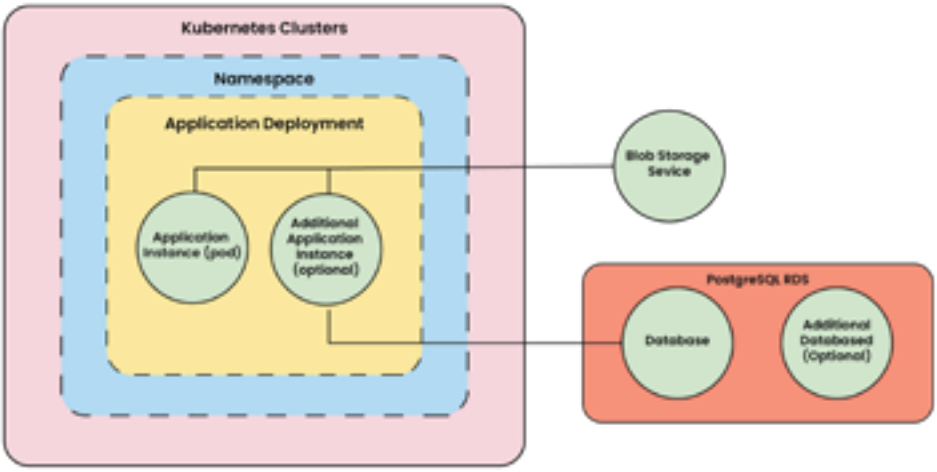
An application environment requires at minimum the following resources:

- 1 PostgreSQL Database
- 1 Document Storage Container
- 1 Kubernetes Cluster (on at least one Node)

Applications are built and deployed using Docker Containers, a tool which packages application build with system level dependencies, such that the application can run independent on its host system.

Applications are stateless, meaning any number of container deployments can share access to the database and blob storage. If multiple containers are put behind one load balancer, we can scale resources as needed. We use Kubernetes to run containers and manage load-balancing.WorkingMouse utilises Docker containers to maximise scalability and flexibility. The Docker containers have the added benefit of easily being able to maintain an up to date and patched operating environment.

WorkingMouse utilises containers released by Microsoft for the purpose of minimising the risk associated with outdated Docker containers. Docker containers can also be independently scanned for vulnerabilities using scanning tools provided by Docker. The default Docker images used for deployment are produced by Codebots to maximise standardisation and minimise the risk of misconfiguration.



Network Architecture

Common Network mitigations - Firewall

Firewalls can be used to protect web apps by filtering, monitoring, and blocking any malicious HTTP/S traffic travelling to the web application, and prevents any unauthorised data from leaving the app. It does this by adhering to a set of policies that help determine what traffic is malicious and what traffic is safe. Just as a proxy server acts as an intermediary to protect the identity of a client, a web application firewall operates in the reverse, acting as an intermediary that protects the web app server from a potentially malicious client.

Some firewalls that are available include:

Azure Network Firewall

Azure Firewall is a cloud-native and intelligent network firewall security service that provides the best of breed threat protection for your cloud workloads running in Azure. It's a fully stateful, firewall as a service with built-in high availability and unrestricted cloud scalability. It provides both east-west and north-south traffic inspection. With a standard firewall, it provides L3-L7 filtering and threat intelligence feeds directly from Microsoft Cybersecurity. Threat intelligence-based filtering can alert and deny traffic from/to known malicious IP addresses and domains, which are updated in real-time to protect against new and emerging attacks.

Layer 3 firewalls (Network firewalls) filter traffic based on the TCP/IP stack. On this layer, traffic can be categorised according to IP address, port numbers and service protocols. By configuring network traffic filtering rules, it can be configured to whitelist or blacklist certain IP addresses. IP whitelisting allows to only provides access from specific IP addresses. At the same time, it blocks access for computers attempting unauthorised access from all unspecified IP addresses. When a new mine site wants to use the app, our DevOps would have to add their site IP address to the list. This could be inconvenient if a user wants to operate outside the approved network.

Read more:

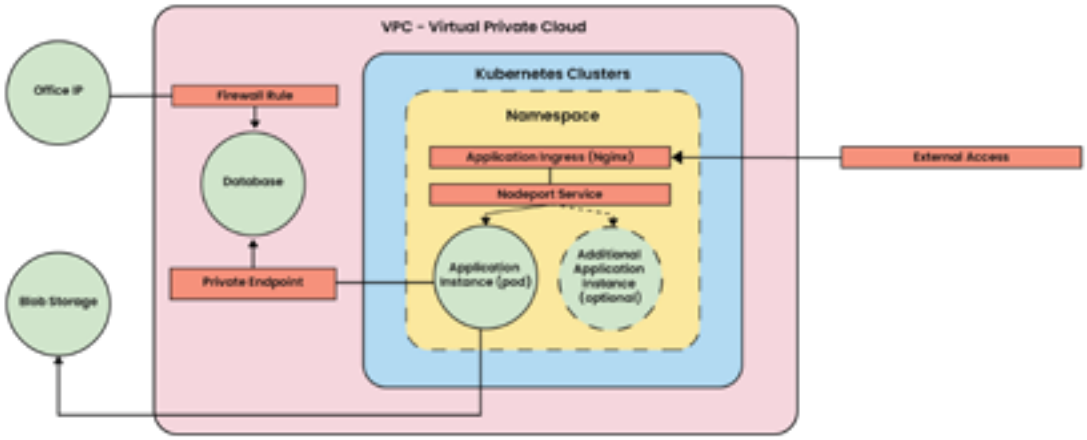
- Azure Firewall Standard features

Azure Web Application Firewall

Web Application Firewall (WAF) provides centralised protection of your web applications from common exploits and vulnerabilities. WAF can defend against malicious attacks that exploit commonly known vulnerabilities such as SQL injection and cross-site scripting. WAF can be deployed with Azure Application Gateway, Azure Front Door, and Azure Content Delivery Network service.

Read more:

- What is Azure Web Application Firewall on Azure Application Gateway?
- Azure Web Application Firewall on Azure Front Door
- Azure Web Application Firewall on Azure Content Delivery Network from Microsoft



Network Components:

- Virtual Network
- Firewall/Security Group rules
  - Allow external traffic into DB
  - Internal office IP
- Cluster - Contained within the virtual network

The network configuration has some key components that relate to security:

- VPC - Virtual Private Cloud
  - A VPC is used to ensure that all components inside the cloud can communicate as required, but no external access is provided outside of the prescribed ingress points defined for the application. In the diagram above, this can be seen as the following:
    - Office IP/VPN - allows access to the internal VPC components for administration and management purposes but only from internal office network or via a VPN.
    - End user access via the Nginx web server. This traffic is encrypted using HTTPS to prevent man in the middle attacks and data from being compromised.
- Blob storage is protected from external access via key/origin based access controls.

Data Integrity and Security

Blob storage providers often come with a suite of features to guarantee data integrity and security.

Controls are in place within the Azure storage product to ensure data integrity and security. This includes encryption, soft deletion, and hashing. Other features include the ability to enable “soft deletion”. When a file is deleted, it can be restored up to a set amount of time. This is a powerful feature for business critical files.

There is a white paper titled Protecting Data in Microsoft Azure published by Microsoft that provides even greater insight into data security and the many controls and default configurations put in place by Microsoft to ensure that stored data is protected.

Within the application context itself, file ownership is secured using the same mechanisms used to protect endpoints from unauthorised access.

Quick Facts	
Redundancy	Redundancy protects against data corruption, as well as offering backups should services go down. By default, we select Locally Redundant Storage for pre-production storage accounts, which creates 3 local copies to prevent data corruption. By default, production storage accounts use Globally Redundant storage, which additionally saves a copy in a different region. This can be another Australian region if required. More information of redundancy types can be found in official documentation: Azure Storage redundancy
Encryption	Encryption on-disk is on by default. Azure offers the ability to double-encrypt. You can enforce TLS levels for encryption in transit. TLS 1.2 is widely supported, but TLS 1.3 is most secure (at time of writing).

Resource Isolation

Azure allows running applications and virtual machines (VMs) on shared physical infrastructure. One of the prime economic motivations for running applications in a cloud environment is the ability to distribute the cost of shared resources among multiple customers. This practice of multi-tenancy improves efficiency by multiplexing resources among disparate customers at low costs.

Option 1: Complete Resource Isolation

For clients who want complete isolation, it is possible to spin up a new instance by allocating their own dedicated resources. This would be the simplest solution to create an effective level of isolation. However, this would result in higher running costs as resources won't be shared.

Option 2: Co Tenanting

There are multiple levels of resource sharing or isolation that can be implemented in this architecture, and it would depend on the requirements of the application. For example, there could be one data server with each client having their own database. This would require a higher initial setup cost but would result in lower running costs compared to option 1.

On Premise

On premise deployments utilise a customer's own hardware, usually installed onsite to execute the application. This hardware is maintained and managed by the customer and may be connected to the web and/or a local network.

There are still vulnerabilities with this option, most of which are also present in desktop applications.

Common Mitigations

For web-based applications, there are some common mitigations that can be put in place from a DevOps perspective that can greatly reduce the attack plane for an application while still allowing for the benefits of a cloud-based application to be maintained. These mitigations are in addition to the security supplied by the application.

VPN

This allows for the benefits of the web, namely in terms of access, to be capitalised upon while ensuring that only a selected set of users can gain access to it. This is a multi-layered approach to security, as users will still need to have access to valid credentials and be authorised to utilise the application even with access to a valid VPN.

Client Certificate

Client certificates are used to concretely identify and validate individual users. If a server is enabled with client certificate authentication, only users who attempt to connect from a client-side loaded with the right client certificates will succeed. A client certificate is sent from the client-side to the server at the start of a session and is used by the server to authenticate the client-side. Even if a legitimate user attempts to connect with the right username and password, if that user isn't on a client application loaded with the right client certificate, that user will not be granted access. Without that certificate, a user won't even be able to see the login page. On top of two-factor authentication (Username and Password, Authenticator app), client certificate authentication could be added as a third layer of authentication.

IP Whitelist

IP whitelist (passlist, accesslist) access control is an additional option to boost security levels.

Access to the primary application ingress can be based upon an IP whitelisting mechanism where access is granted only for those who are accessing from an approved IP. This would require a static IP address to be configured for a given user, which is usually already in place for many businesses and organisations.

Access Control

Access control can be considered to exist in two parts. The first part being the logical constraints that prevent users from accessing information that they should not, and the second is the principles in use to decide upon these rules.

WorkingMouse recommends following the principle of least privilege in all our access control decisions.

> *Verify enforcement of the principle of least privilege in functions, data files, URLs, controllers, services, and other resources. This implies protection against spoofing and elevation of privilege.*
>
> *- ASVS v4.0.2-1.4.3*

Internal System Administration

- External (Internet) direct SSH access to any server is forbidden except in emergency cases.
- No SSH access via passwords is allowed under any circumstances. Only private/public key pairs are used.
- WorkingMouse suggests using a two-layer system where access to the internal server network can only be gained via an encrypted VPN and access to specific servers can only be gained by possessing a specific SSH key.

Site Resource Access

- HTTPS is a data transfer protocol that encrypts all data transfer between the server and client, client and server. All customer specific data is transferred using HTTPS.
- The application will be hosted in the application owner's account with the cloud provider of their choice, resulting in resource isolation between application owner's applications and other clients.
- Resources cannot be accessed arbitrarily. They can only be accessed according to the strict rules of the site's underlying application.

File/Media Storage

- All uploads to the application are stored in the Blob Storage Service provided by your chosen Cloud Provider.
- Access to the data stored in the Blob Storage Service is tailored to the need of your application when setting up your environment. Efforts are taken to ensure access to this data is as controlled as it needs to be. The default configuration is that the application maintains full access to the blob storage. It then manages access via the security model and custom security written into the app.

Firewall

- All servers are fire walled with only the minimum set of services exposed to allow the server to fulfil its tasks.

Database System Access

- All databases are password protected with several levels of user privileges.
- Databases are only accessible from within the internal private fire walled network. At no time are they exposed to the Internet.
- Only the servers that need database access are granted access
- Applications are given the least database privileges necessary to run the application. For example, a running application does not have permissions to alter the database schema.

Backups

- Database PITR (Point In Time Recovery) is enabled which allows restoring to a specific moment within the last 5 minutes.
- Full database backups are automatically created daily and kept for the last 14 days.
- Blob Storage is covered by the redundancy and availability options provided by your Cloud Provider.
- If desired, Blob Storage backups can be setup.

Endpoint Security

Endpoint security refers to securing the computers that are used to access a service. This is important because even if a service is perfectly secure, it can still be compromised by a malicious actor if access is gained through valid means. Valid means within the context of an application differs depending on the application but for example could be via the computer of a user that has access to the application using a valid set of credentials.

Vulnerabilities

Malware

Malware is easily introduced through URL links in emails or through compromised hardware such as USB sticks. Other more sophisticated methods also exist. Once malware is installed on an endpoint, there is often very little that it cannot do.

It is important to note that desktop applications are even more susceptible to malware-based attacks than web-based applications. In the context of a cloud application, the malware is limited to the level of access of that user. For desktop applications however, the user will generally have a much greater level of access to execute the program. This means there is a wider risk of vulnerabilities in the application.

Desktop applications are also highly susceptible to ransomware attacks, which can perform a denial of service and prevent access to key data. To mitigate ransomware attacks, regular backups are required, which are not always well managed on local machines.

Data Integrity

Web based applications are less susceptible to data integrity issues due to validation and verification being able to be maintained and managed in a controlled environment (the cloud).

For desktop applications, corruption and data loss are more likely due to user error or even hardware failure. Redundancy and data integrity checks are standard for web-based applications but are often lacking in a user's personal workstation.

Managing Endpoint Security

An endpoint protection platform (EPP) is an integrated suite of endpoint protection technologies—such as antivirus, data encryption, intrusion prevention, and data loss prevention—that detects and stops a variety of threats at the endpoint. EPPs exist to provide one-stop shops for end point protection. These services are recommended for all those that are concerned about security, regardless of whether an application is web-based, or desktop based.

Glossary

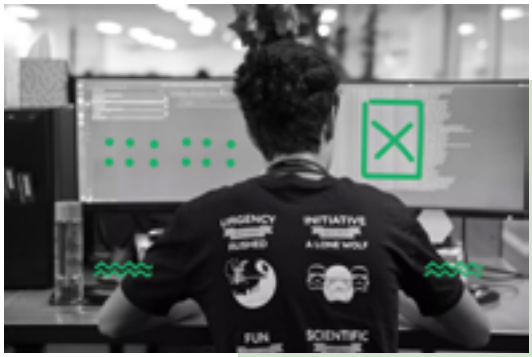
Term	Definition
OWASP	The Open Web Application Security Project (OWASP) is a non-profit foundation dedicated to improving the security of software. In short, OWASP is a repository of all things web-application-security, backed by the extensive knowledge and experience of its open community contributors.
ASVS	The OWASP Application Security Verification Standard (ASVS) Project provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development. The ASVS is a list of application security requirements or tests that can be used by architects, developers, testers, security professionals, tool vendors, and consumers to define, build, test and verify secure applications. The primary aim of ASVS is to normalise the range in the coverage and level of rigour available in the market when it comes to performing Web application security verification using a commercially workable open standard. The standard provides a basis for testing application technical security controls, as well as any technical security controls in the environment, that are relied on to protect against vulnerabilities such as Cross-Site Scripting (XSS) and SQL injection. This standard can be used to establish a level of confidence in the security of web applications.
CWE	A Common weakness enumeration (CWE) is a common security weakness in software or hardware that is often used as a common language for identifying and mitigating security vulnerabilities. A registry of CWEs can be found at https://cwe.mitre.org/.
CVE	A Common Vulnerability and Exposure (CVE) is a publicly disclosed cybersecurity vulnerability or flaw. A list of CVE's can be found at https://cve.mitre.org/.

SAST	Static application security testing. This term stands for the practice of running static analysis tests against source code to uncover security vulnerabilities.
DAST	Dynamic application security testing. This term stands for the practice of running dynamic analysis tests against a running application to uncover security vulnerabilities.
ACL	Access control list. An ACL is the primary contributor to the authorisation of users against given resources.
Codebots	A development tool used to speed up development by providing bots that can develop most of the standard application requirements. https://codebots.com
C#Bot	A custom codebot that writes to a C# .Net Core and React web application stack.

Who Are We?

We're an Australian software company that excel at designing & developing software solutions with the right mix of people, process & tools.

How we can help...



Application Development

Use one of our squads to build your web-responsive and mobile apps - leverage a cross-functional team to design and develop the very best software solution.

Product Design

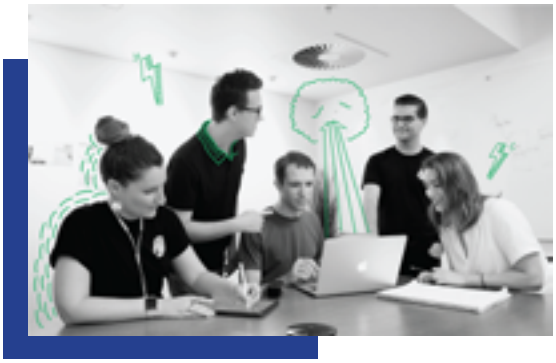
We take a problem-lead approach to design so you build the right thing the first time. A great user interface (UI) and exceptional user experience (UX) ensure application success.

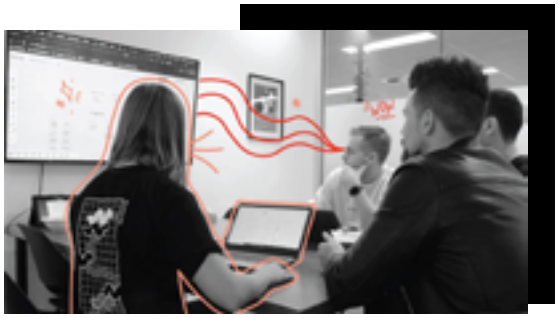
Team Augmentation

Need more development 'oomph'? We have squad leads, designers, software and DevOps developers and solution architects ready to step in and help out.

DevOps Consulting

Stay ahead of the digital transformation game. DevOps empowers and streamlines your projects for faster delivery cycles. Let us guide you into the future.



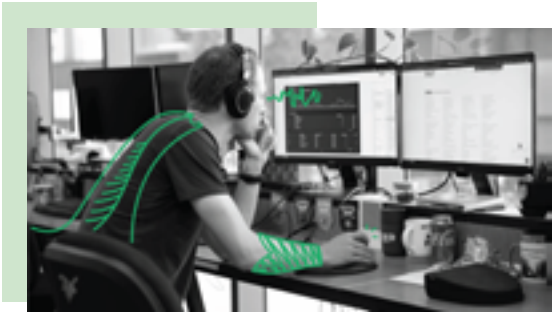


Modernisation Strategy

Connect your organisational goals to your software systems direction with a comprehensive strategy. Take back control and escape the legacy trap.

Product Success

Improve your applications return on investment (ROI). Continuously iterate and improve your product through data-driven insights and metrics.

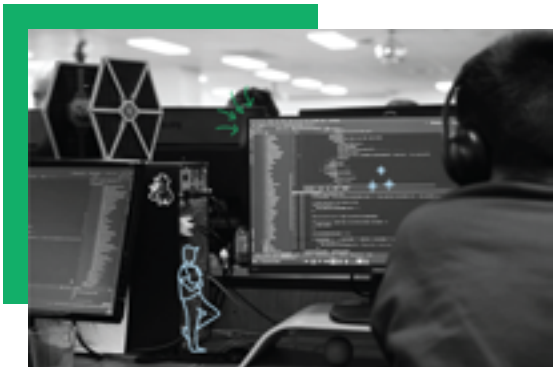


Legacy Projects

Need support while modernising your legacy app? You can rely on our team to keep everything running smoothly while we figure out the next steps.

Development Automation

Strapped for time and resources? Automate parts of your development to free up your dev team to use their talents and creativity where they're truly needed.



workingmouse

Get in touch

Interested in learning more about software or think we might be a good fit for your next project?

(07) 3606 0230

growth@workingmouse.com.au