Microsoft

Secure DevOps: Application Security Principles and Practices

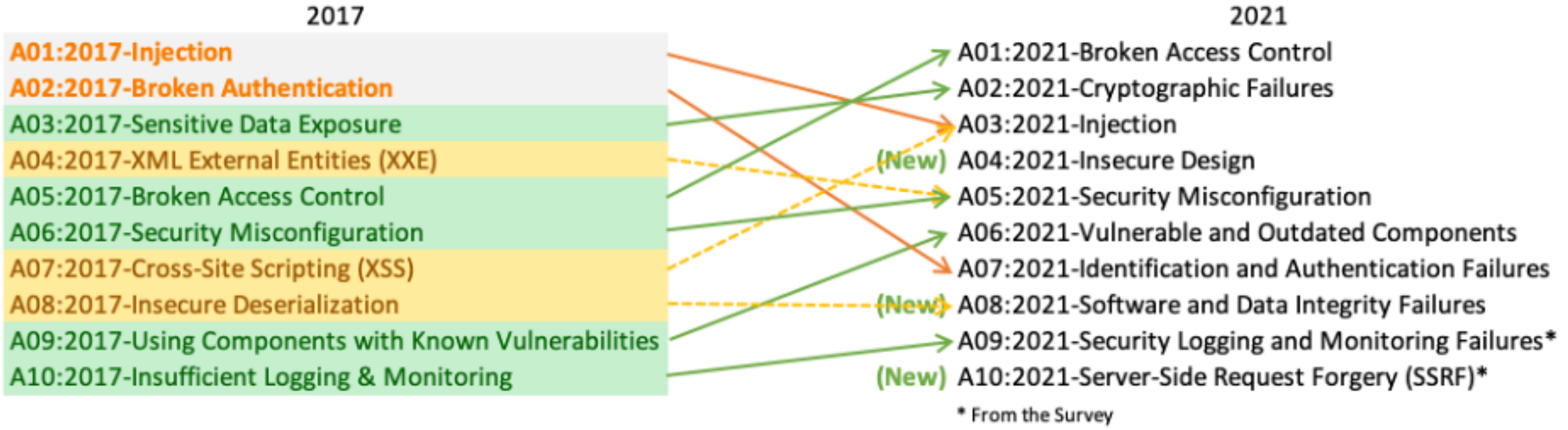
Module 9:
OWASP Top 10 2021

Microsoft Services

Lesson: OWASP Top Ten

OWASP TOP 10

Open Web Application Security Project



Top Ten in 2021

Open Web Application Security Project

TOP 10

- A01:2021 - Broken Access Control
- A02:2021 - Cryptographic Failures
- A03:2021 - Injection
- A04:2021 - Insecure Design
- A05:2021 - Security Misconfiguration
- A06:2021 - Vulnerable and Outdated Components
- A07:2021 - Identification and Authentication Failures
- A08:2021 - Software and Data Integrity Failures
- A09:2021 - Security Logging and Monitoring Failures
- A10:2021 - Server-Side Request Forgery

A01:2021 - Broken Access Control

Occurs when confidential information is viewed by a user who should not have permission to access that data

✅ DO 

- Authorize users on all externally facing endpoints. The .NET framework has many ways to authorize a user
- Set a cookie policy
- Set secure password policy
- Understand system's trust boundaries

❌ DON'T 

- Roll your own authentication or session management, use the one provided by .NET

A01:2021 - Broken Access Control

ASP.NET Attributes

```
public class AccountController : Controller
{
  public ActionResult Login()
  {
  }

  public ActionResult Logout()
  {
  }
}
```

```
[Authorize(Policy="", Roles="")]
public class AccountController : Controller
{
   [AllowAnonymous]
   public ActionResult Login()
   {
   }

   public ActionResult Logout()
   {
   }
}
```

A01:2021-Broken Access Control

Minimal API

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddAuthorization(o => o.AddPolicy("AdminsOnly", b =>
b.RequireClaim("admin", "true")));

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");

builder.Services.AddDbContext<ApplicationDbContext>(options => options.UseSqlServer(connectionString));

builder.Services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
                .AddEntityFrameworkStores<ApplicationDbContext>();

var app = builder.Build();

app.UseAuthorization();

app.MapGet("/auth", [Authorize] () => "This endpoint requires authorization.");

app.MapGet("/", () => "This endpoint doesn't require authorization.");

app.Run();
```

A01:2021 - Broken Access Control

ASP.NET Blazor

```
<AuthorizeView Roles=“admin, superuser” Policy="content-editor">
        <Authorized>
                <h1>Hello, @context.User.Identity.Name!</h1>
                <p>You can only see this content if you're authorized.</p>
                <button @onclick="SecureMethod">Authorized Only Button</button>
        </Authorized>
        <NotAuthorized>
                <h1>Authentication Failure!</h1>
                <p>You're not signed in.</p>
        </NotAuthorized>
</AuthorizeView>

@code {
    private void SecureMethod()
    {
    }
}
```

A02:2021 - Cryptographic Failures

Failures related to cryptography that often leads to sensitive data exposure or system compromise

✅ DO 

- Classify the data
- Protection data in transit and at rest and in use (Azure confidential computing)
- Use strong cryptographic algorithms like AES

❌ DON'T 

- Store data if you don't have to
- Write your own cryptographic libraries

A02:2021 - Cryptographic Failures

Use System.Security.Cryptography

```
Aes aes = Aes.Create();
CryptoStream cryptStream = new CryptoStream(fileStream,
                                            aes.CreateEncryptor(aes.Key, aes.VI),
                                            CryptoStreamMode.Write);
```

```
public static byte[] HashPassword256(string password)
{
     System.Security.Cryptography.SHA256 mySHA256 = System.Security.Cryptography.SHA256.Create();
     var encoding = new System.Text.UnicodeEncoding();
     return mySHA256.ComputeHash(encoding.GetBytes(password));
}
```

```
var randomNumberGenerator = System.Security.Cryptography.RandomNumberGenerator.Create();
```

https://docs.microsoft.com/en-us/dotnet/standard/security/encrypting-data

A03:2021 - Injection

Happens when an application accepts data as input and processes it as instruction instead of as data

✅ DO 

- Validate on Client and Server side
- Neutralize or verify user input in your web application

❌ DON'T 

- Assume any user data input as safe to process

A03:2021 - Injection

Build-in capabilities

```
using System.ComponentModel.DataAnnotations;

public class ExampleModel
{
        [Required]
        [StringLength(10, ErrorMessage = "Name is too long.")]
        public string? Name { get; set; }

        [Required]
        [Range(typeof(bool), "true", "true", ErrorMessage = “Unapproved design.")]
        public bool IsValidatedDesign { get; set; }
}
```

A03:2021 - Injection

ASP.NET Blazor File Upload

```
<InputFile OnChange = "@LoadFile" />

@code {

    private string[] permittedExtensions = { ".txt", ".pdf" };

    private void LoadFile(InputFileChangeEventArgs e)
    {
        var ext = Path.GetExtension(e.File.Name).ToLowerInvariant();

        if (string.IsNullOrEmpty(ext) || !permittedExtensions.Contains(ext))
        {

        }
    }
}
```

A03:2021 - Injection

ASP.NET Blazor File Upload

1. `var reader = await new StreamReader(browserFile.OpenReadStream()).ReadToEndAsync();`

2.
```
var memoryStream = new MemoryStream();
browserFile.OpenReadStream().CopyToAsync(memoryStream);
await blobContainerClient.UploadBlobAsync(trustedFilename, memoryStream));
```

3.
```
await using FileStream fs = new(path, FileMode.Create);
await browserFile.OpenReadStream().CopyToAsync(fs);
```

4. `await blobContainerClient.UploadBlobAsync(trustedFilename, browserFile.OpenReadStream());`

A03:2021 - Injection

B I Y F

> ⚠ **Warning**
>
> **Don't trust file names supplied by clients** for:
>
> - Saving the file to a file system or service.
> - Display in UIs that don't encode file names automatically or via developer code.
>
> For more information on security considerations when uploading files to a server, see **Upload files in ASP.NET Core**.

A03:2021 – Injection

Did you really name your son ***Robert'); DROP TABLE STUDENTS;--***?

1. Vulnerable statement

```
string sql =
"SELECT * FROM users WHERE name = '" + username + "';";
```

2. Setting the 'username' to

```
a';DROP TABLE users;--
```

3. Renders the following SQL statement

```
SELECT * FROM Users WHERE name = 'a';DROP TABLE users;--
```

Source: https://www.xkcd.com/327/

A03:2021 - Injection

SQL injection example

1. View item details

https://[URL]/site/Result?id=1

2. Setting the item ID to

```
0 UNION ALL SELECT 0,(SELECT STRING_AGG(table_name,',')),'','','' FROM
information_schema.tables;--
```

3. Once table information is obtained

```
0 UNION ALL SELECT 0,(SELECT
STRING_AGG(CONCAT(Email,'|',PasswordHash),',')),'','','' FROM
[dbo].[AspNetUsers];--
```

A04:2021 - Insecure Design

Addresses design level flaw, others focus more on code-level issues

✅ DO

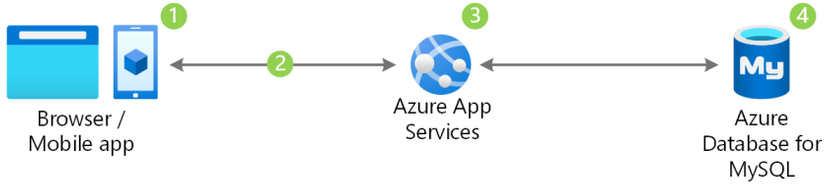
- Compensating Controls
- Least privilege
- Attack surface reduction
- Zero Trust
- Privacy
- Defense in Depth
- Threat Modeling
- Reference architectures

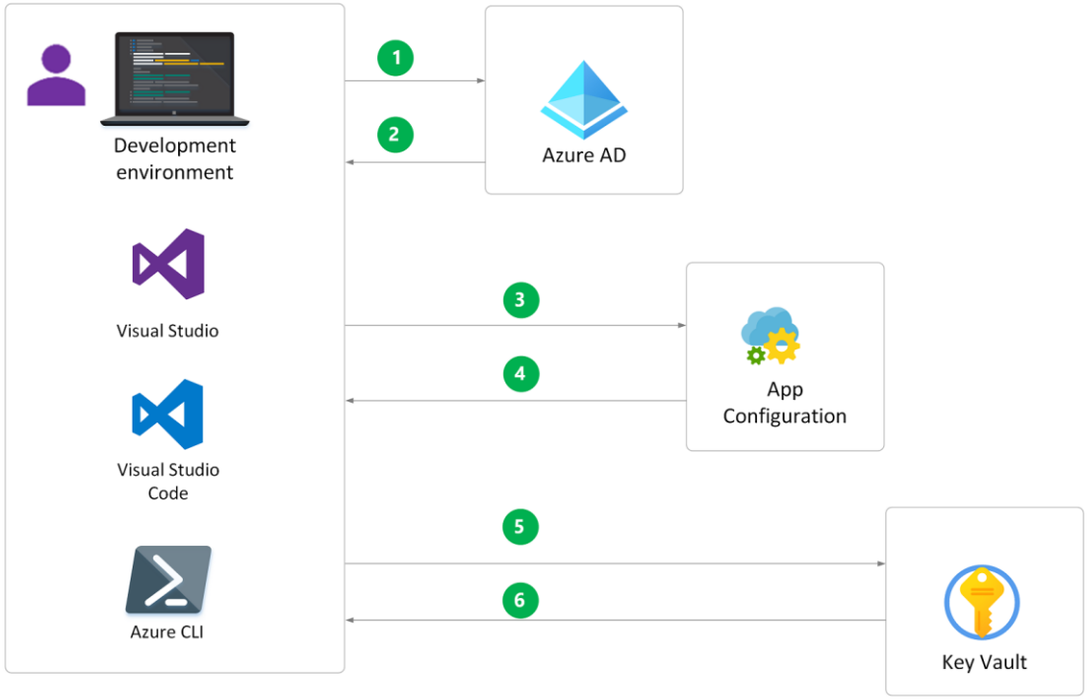
❌ DON'T

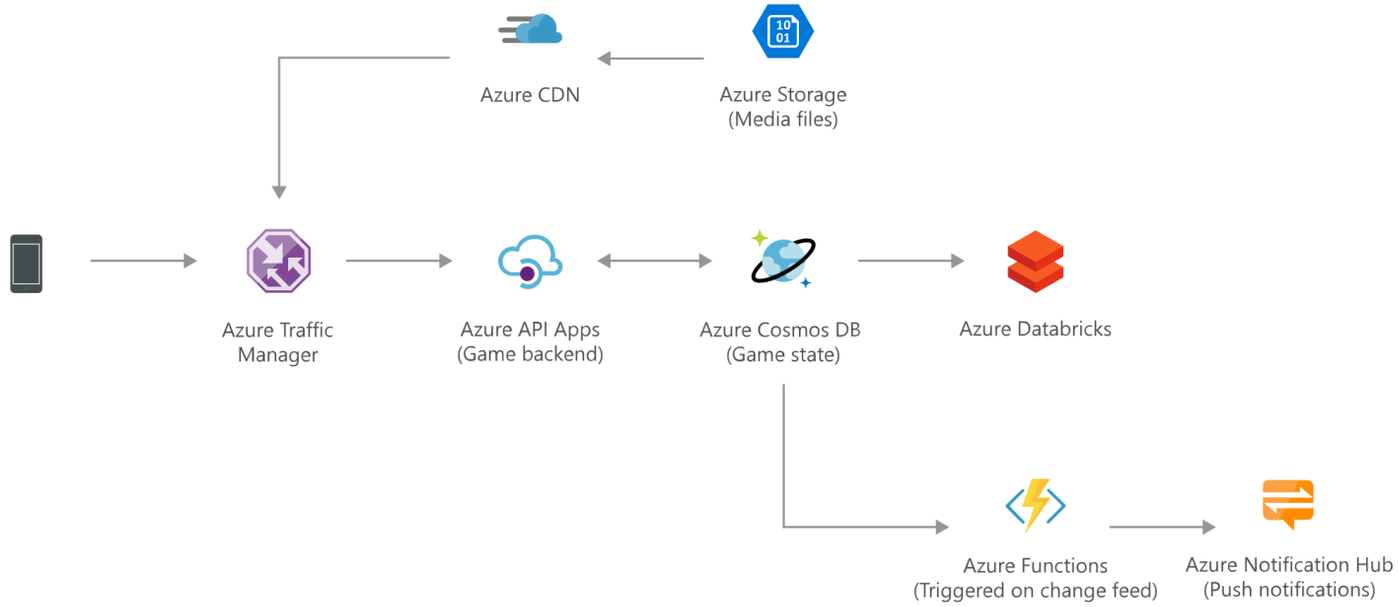
- Store passwords in plain text
- Think of security as an afterthought

A04:2021 - Insecure Design

Reference architecture



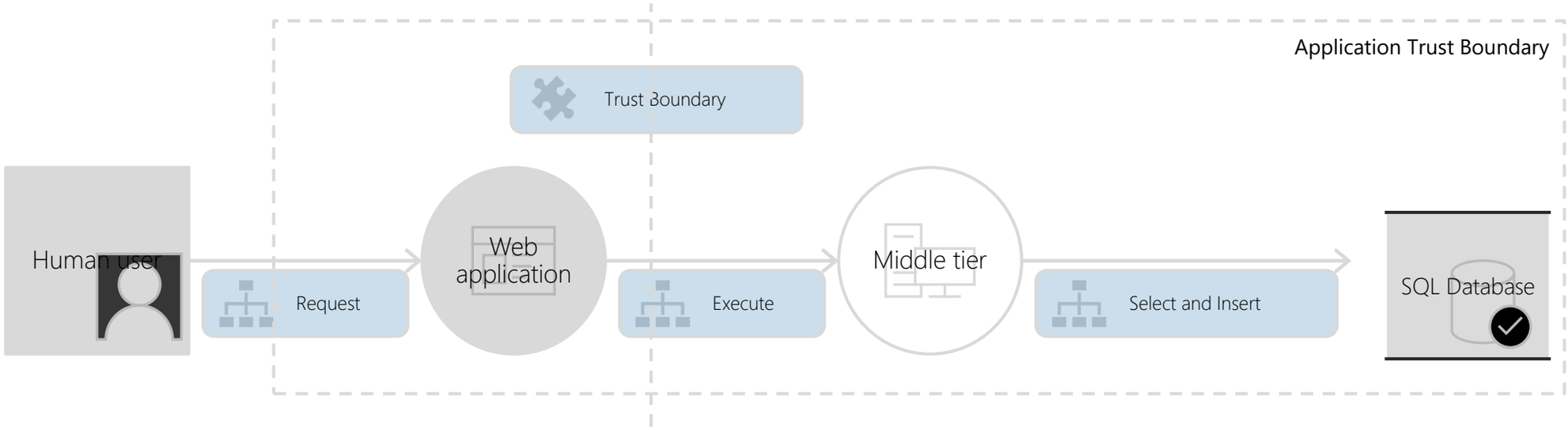




A04:2021 - Insecure Design

Threat Modeling

Process to understand **security threats** to a system, determine **risks** from those threats, and establish appropriate **mitigations**.



A05:2021-Security Misconfiguration

Challenge the defaults

🤘 DO

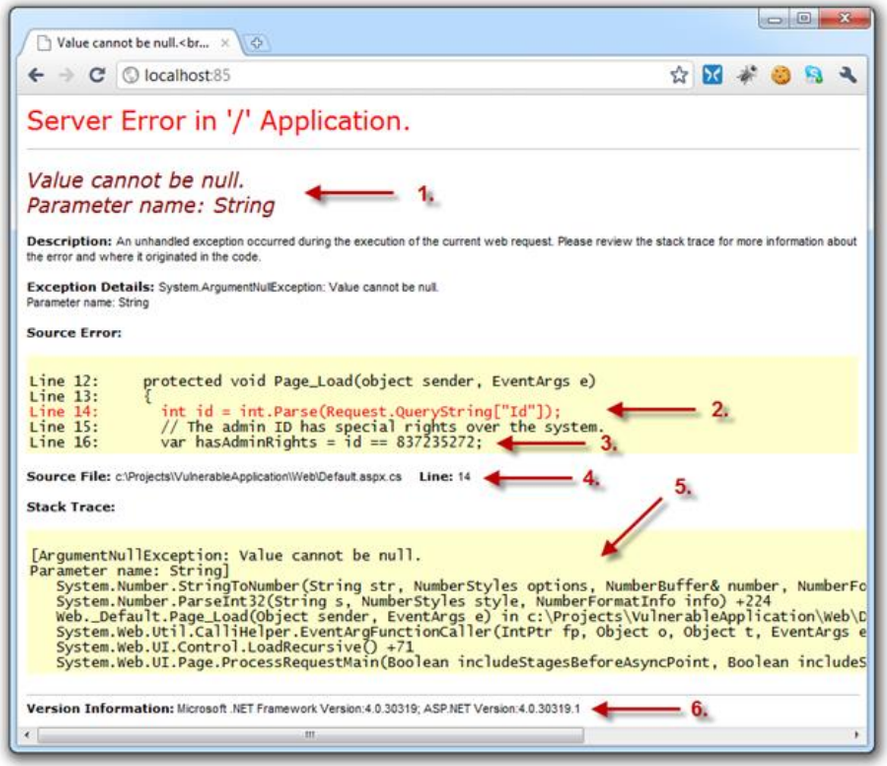
- Change default passwords (and enable MFA)
- Disable unnecessary services or features (YAGNI)
- Use HTTPS

❌DON'T

- Enable unnecessary features, ports, services, pages, accounts, or privileges
- Convenience is the enemy of security

Source: https://www.bonfire.com/dont-accept-the-defaults-abel-wang

A05:2021 - Security Misconfiguration




```
protected void Page_Load(object sender, EventArgs eventArgs)
{
    try
    {
        object o = null;
        o.ToString();
    }
    catch (Exception e)
    {
        this.Response.Write(e.ToString());
    }
}
```

```
protected void Page_Load(object sender, EventArgs eventArgs)
{
    try
    {
        object o = null;
        o.ToString();
    }
    catch (Exception e)
    {
        this.Response.Write("An error occurred. Please try again later.");
    }
}
```

```
<customErrors mode="RemoteOnly" redirectMode="ResponseRewrite" defaultRedirect="~/Error.aspx" />
```

Image Source: https://www.troyhunt.com/owasp-top-10-for-net-developers-part-6

A06:2021 - Vulnerable and Outdated Components

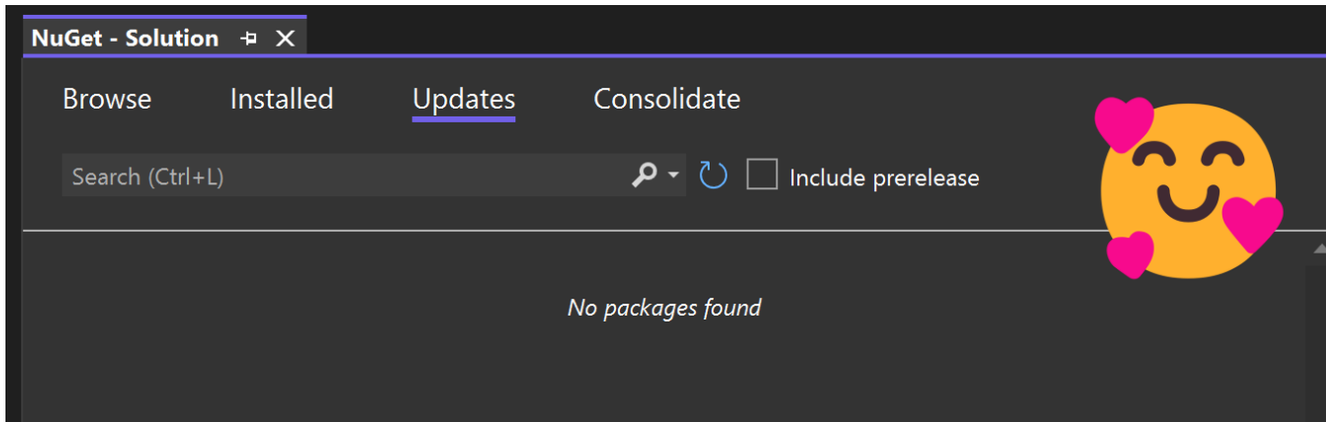
Occurs when you don’t know all the versions of all the components you use and if the software is vulnerable unsupported or out of date

✅ DO

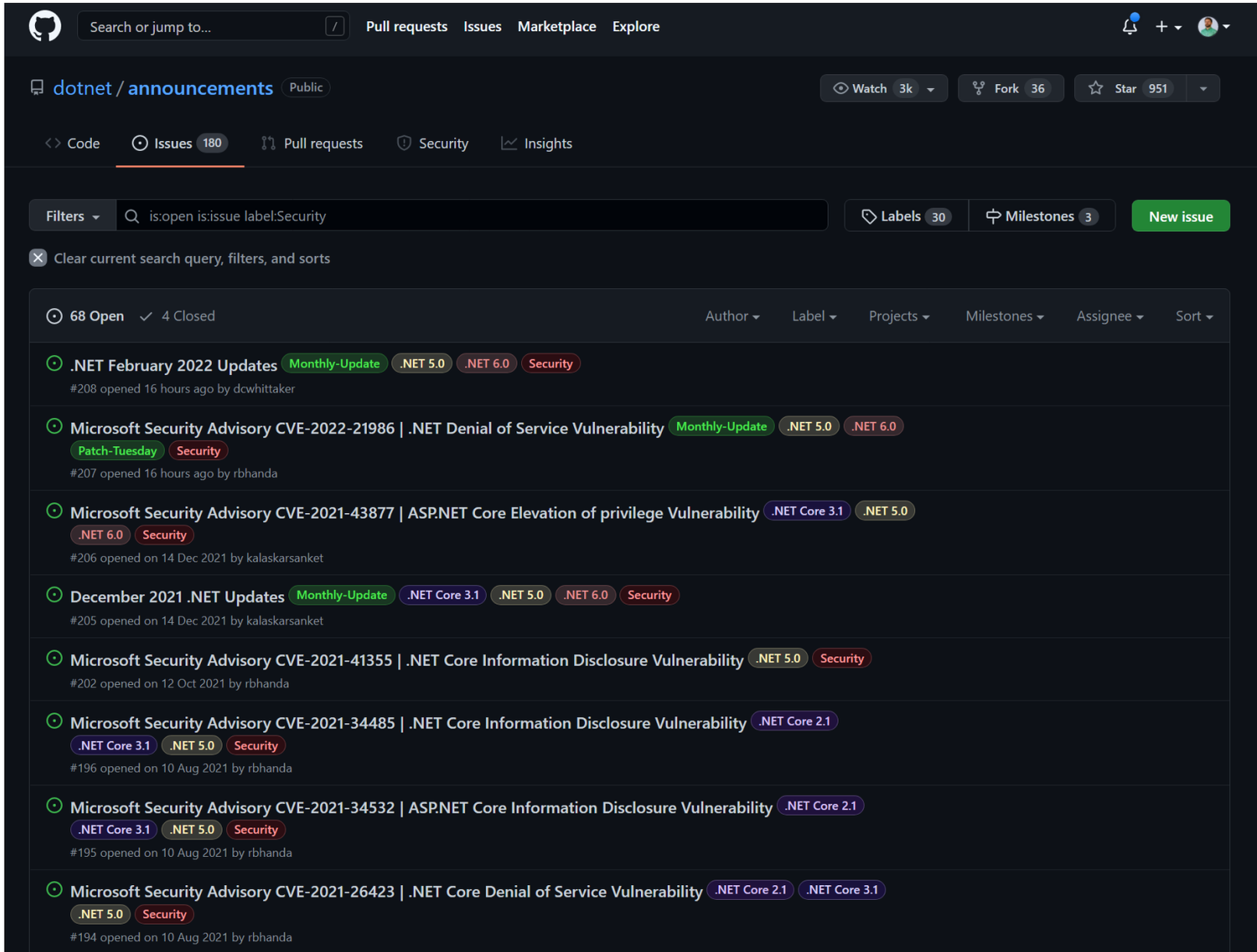
- Choose up-to-date and secure dependencies
- Examine the current vulnerability status of dependencies before shipping
- Continuously review the changing risk inherited post deployment
- Automated SCA Scans

❌ DON’T

- Bring dependencies to components from unknown sources



A06:2021 - Vulnerable and Outdated Components



A07:2021 - Identification and Authentication Failures

Confirmation of the user's identity, authentication and session management is critical to protect against authentication–related attacks

DO

- Implement multi-factor authentication
- Do not ship or deploy with any default credentials, particularly for admin users

DON'T

- Re-use passwords and/or share them across environments

A08:2021 - Software and Data Integrity Failures

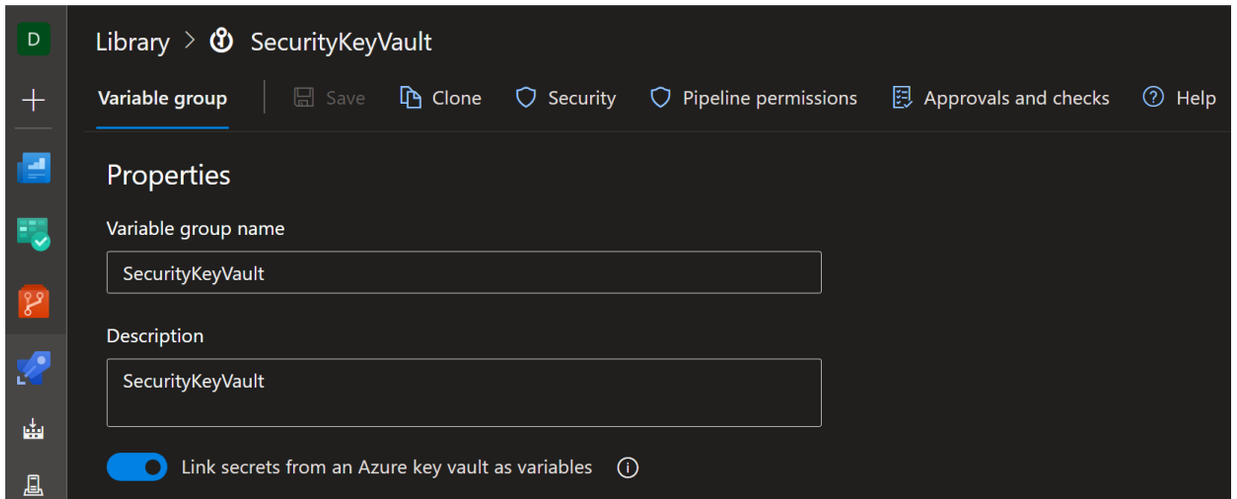
Happens when an application relies upon plugins, libraries, or modules from untrusted sources, repositories

✅ DO

- Only depend on vetted and verified libraries and components
- Secure your CI/CD pipeline

❌ DON'T

- where updates are downloaded without sufficient integrity verification



A09:2021 - Security Logging and Monitoring

Logging and monitoring gives us an opportunity to stop the attacker in their tracks

✅ DO 

- Ensure all login, access control failures and server-side input validation failures can be logged with sufficient user context to identify suspicious or malicious accounts
- Establish effective monitoring and alerting

❌ DON'T 

- Leave your logs unprotected
- Log sensitive data such as user's passwords

Microsoft.Extensions.Logging.ILoggerFactory

A10:2021 - Server-Side Request Forgery

Occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination

✅ DO

- Validate all input
- Deploy defence in depth
- Deny by default

❌ DON'T

- Ignore input validation
- Use HTTP redirections

Lesson: Automation and Tooling

Three themes for successfully securing the developer workflow

Developer – First Tooling

Native and built-in security capabilities

Automation

Secure your code

Find hard-coded secrets in code base

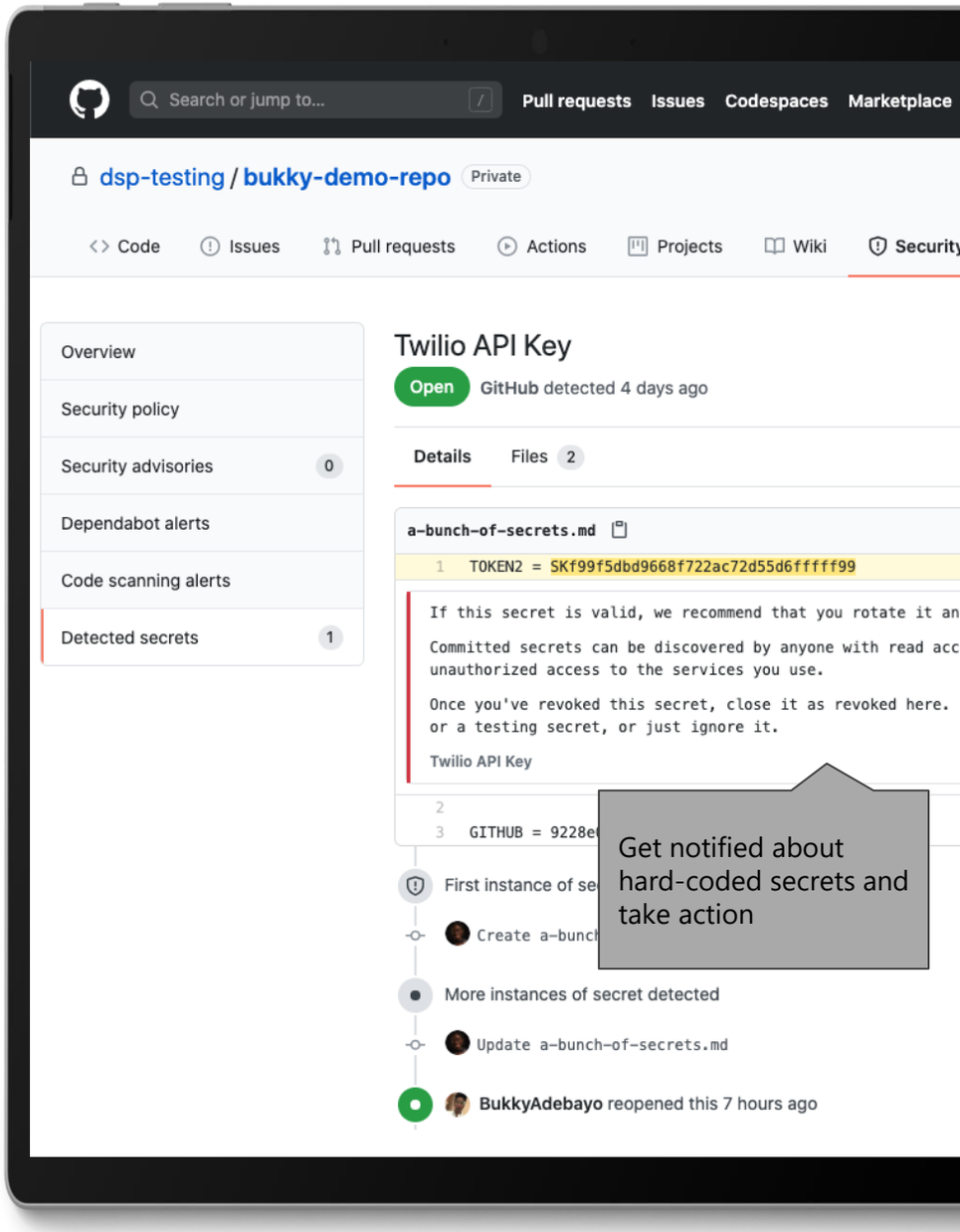
Get access to Vulnerability Dependency Insights and automated security fixes with Dependabot

Pattern-based security analysis

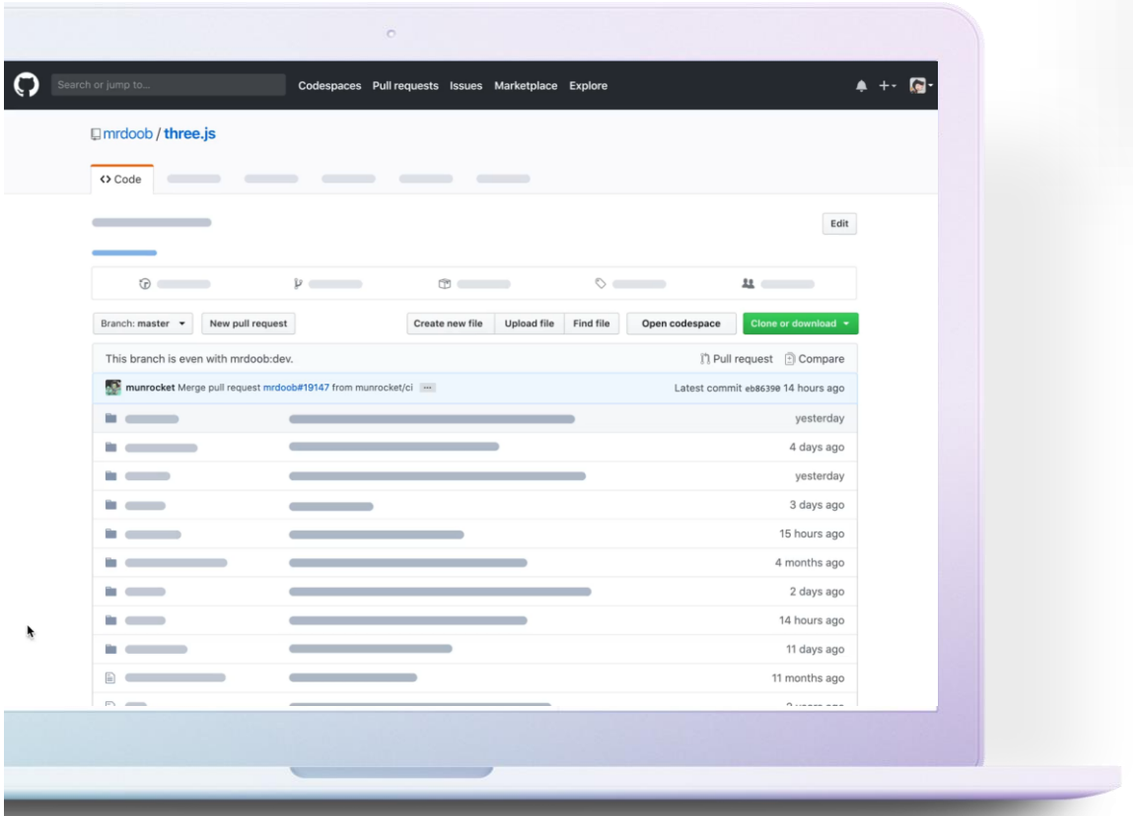
Use security analysis with GitHub advanced security scanning to access both open source repositories and enterprise code

Global community for security

Take advantage of integrations with the National Vulnerability Database, MITRE, and WhiteSource for up-to-date security information



Secure cloud-based developer machines



ENSURE DEVELOPER VELOCITY, SECURELY

For code browsing, limit scope for non-trusted repositories to a browser sandbox

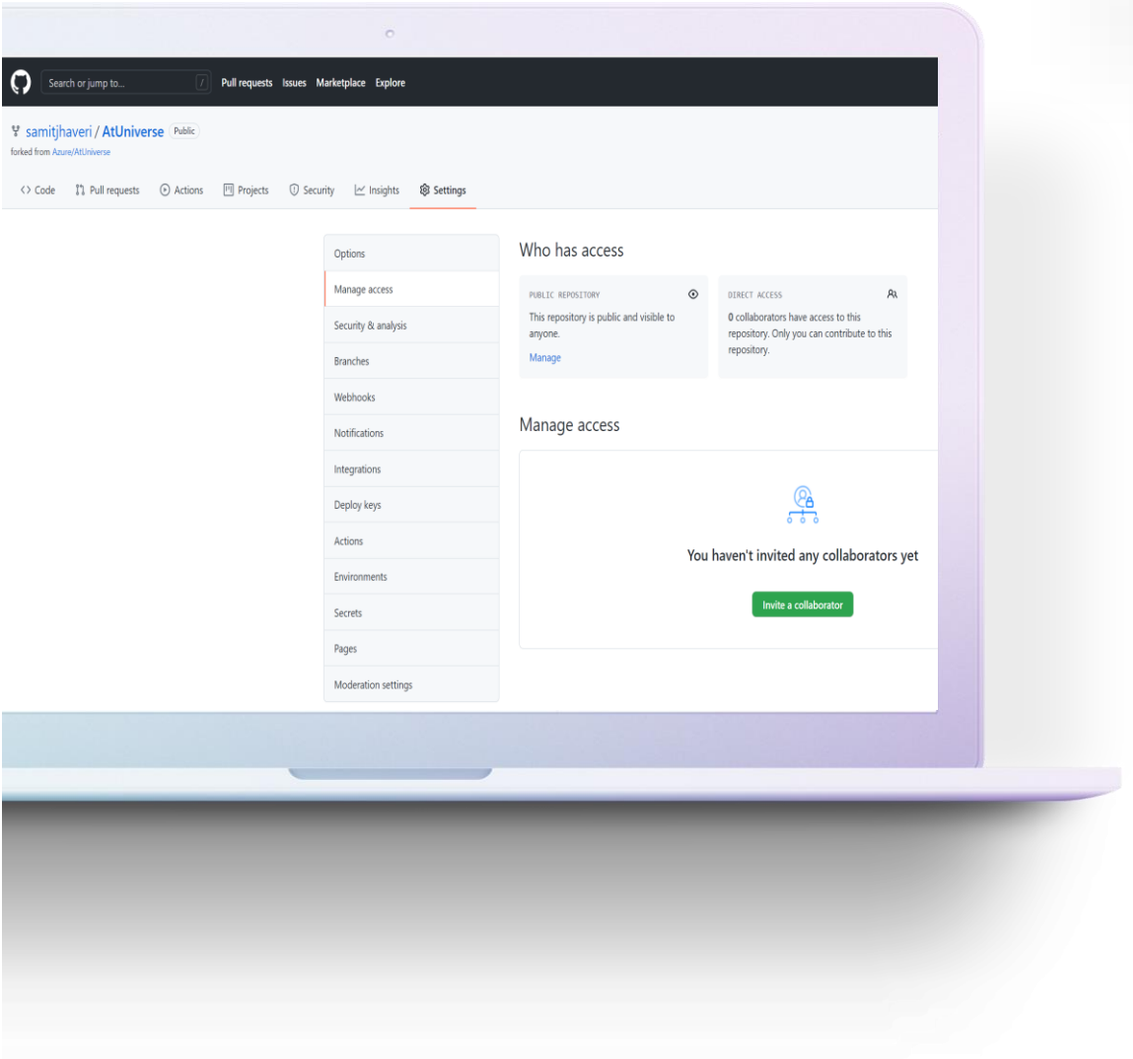
Build non-trusted repositories in an isolated environment not on a local developer machine

Cloned repositories should implement least privileged access principles

PREVENT THESE TYPES OF ATTACKS:

- Ability to execute malicious code on a developer machine
- Compromised developer machines acting as "jump-box" to further systems

Harden access to codebases



Ensure all codebases must have a maintainer

Follow least privileged access - grant access to developers to codebases by organization (no global access)

Limit elevated privileges. Do so only when absolutely necessary and implement a finite duration on higher access levels

PREVENT THESE TYPES OF ATTACKS:

- Source code exfiltration on developer machines
- Source code tampering or injection of malicious code

Secure your dependencies

Manage your dependencies

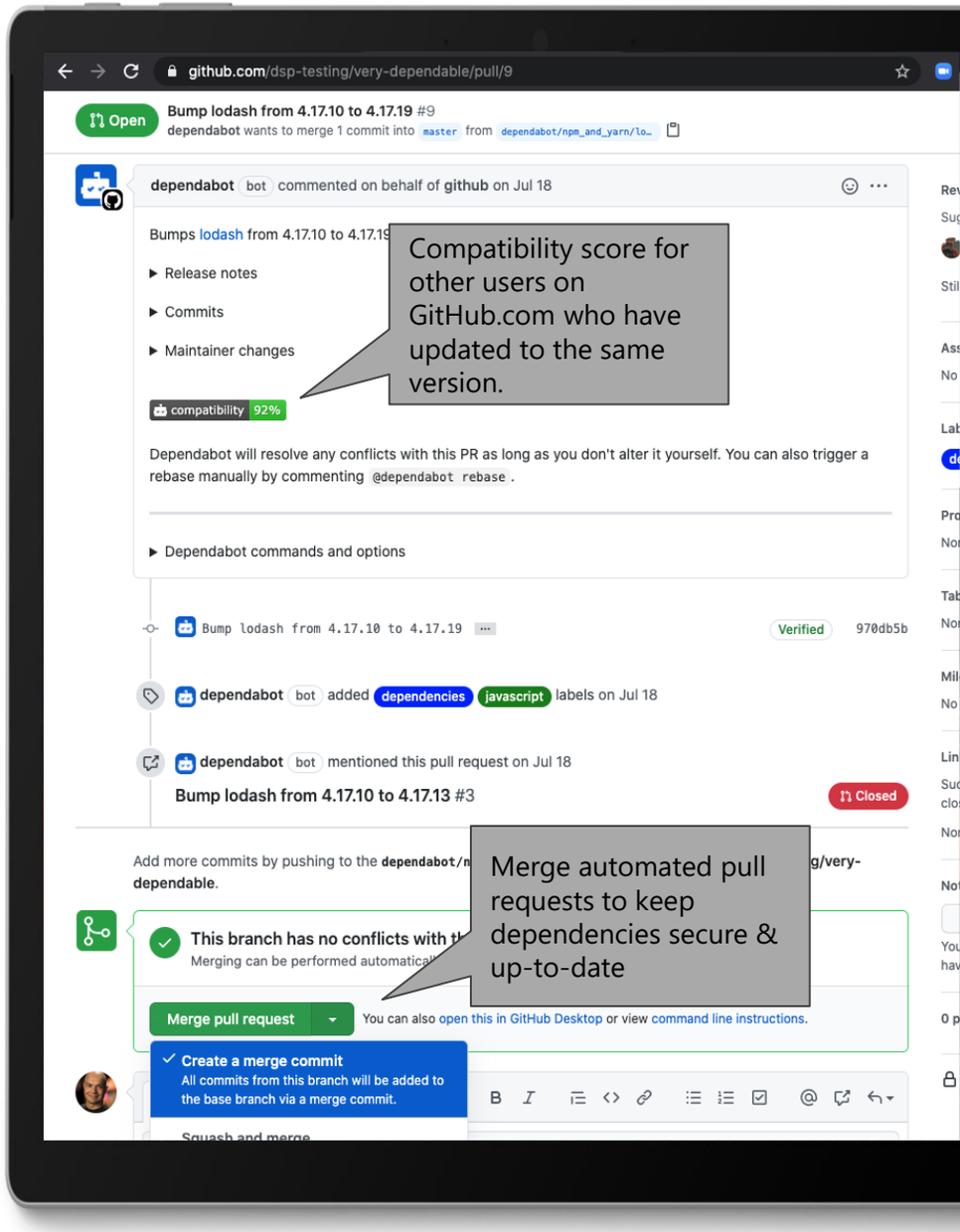
Scan for tokens and secrets across open source, InnerSource and commercial components used in the projects

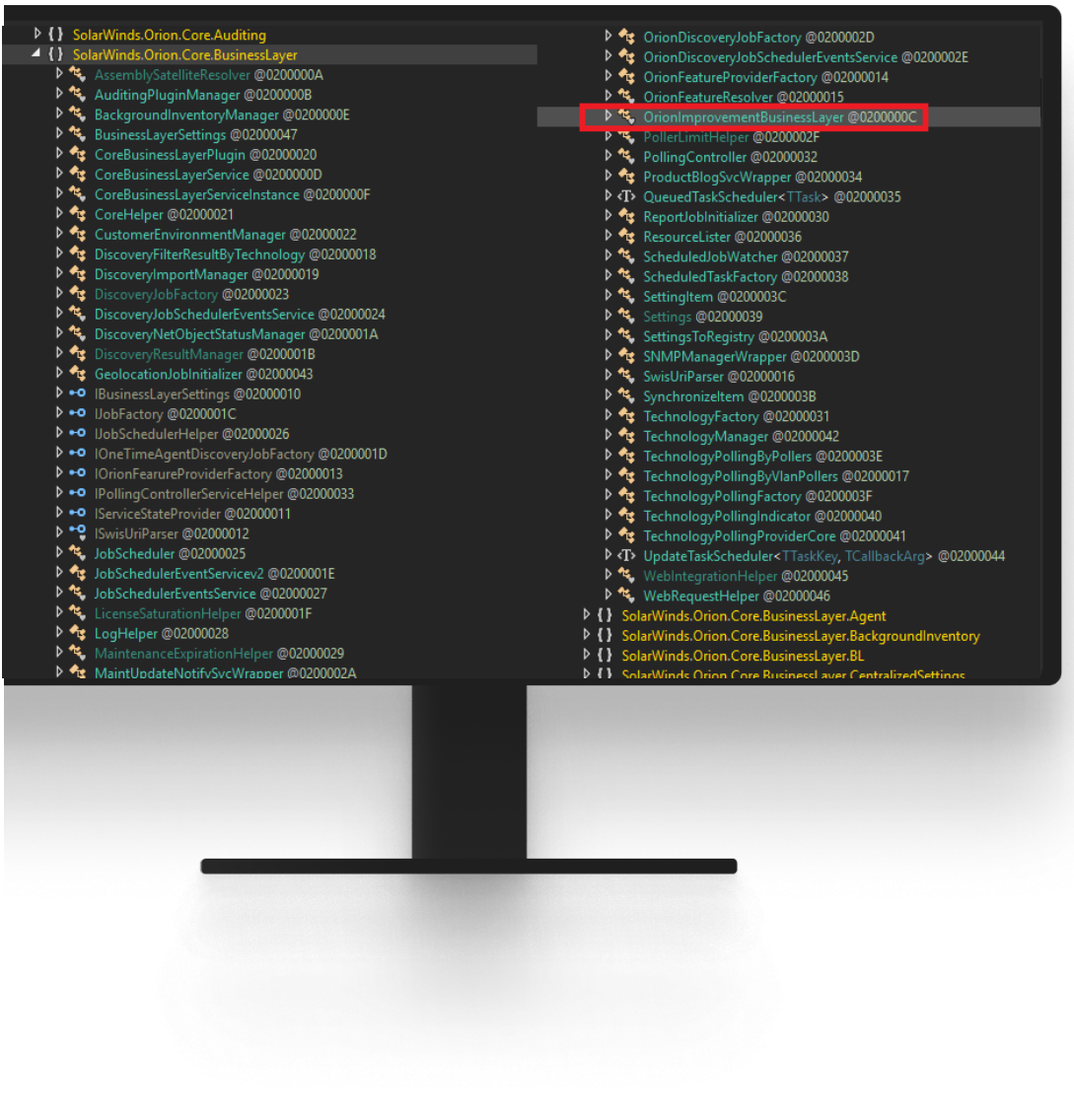
Know your environment

Secure your dependencies including where code is being deployed whether that's on-premises or in the cloud

Fix and publish vulnerability information

Examine your Dependency Graph to see the packages your project depends on and the repositories that depend on it





Secure the DevOps Pipelines

ENABLE DEVELOPER VELOCITY, SECURELY

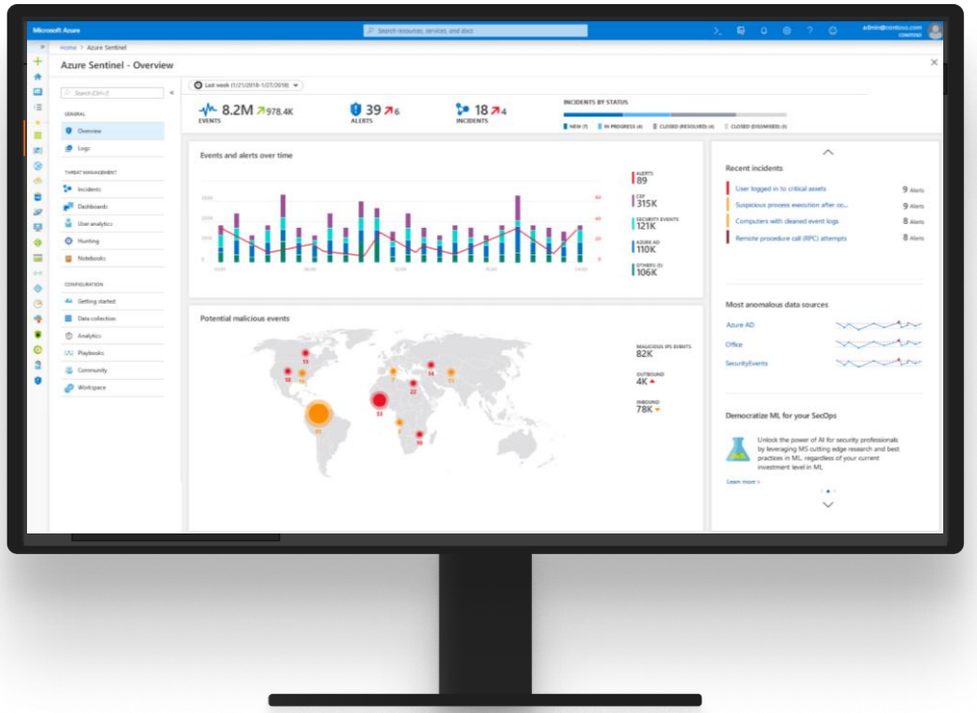
Adopt a centrally governed and secure engineering system

Only allow trusted, approved and authorized DevOps pipelines and actors the ability to enforce supply chain controls

Control deployment from non-trusted pipelines by denying production certs

PREVENT THESE TYPES OF ATTACKS:

- Build tampering
- Unauthorized access to package repository and production infrastructure



Monitor the developer cloud

AUTOMATED PLAYBOOK TO DETECT AND RESPOND TO SUSPICIOUS ACTIVITIES

Monitor for anomalous activities to identify and detect that which may represent nefarious intent from mass repo cloning, downloading, or deletion

Regularly scan for identity access management to ensure least-privileged access management policies

Enable alerting and routing for security policies across DevOps pipelines

Automate notifications to developers of leaked tokens, secrets, or credentials

PREVENTS THESE TYPES OF ATTACKS:

- Threat from within
- Escalation of brute force attack/intrusion access

Secure your workflow with GitHub Actions

GitHub Actions available now:

Orchestrate policy integration

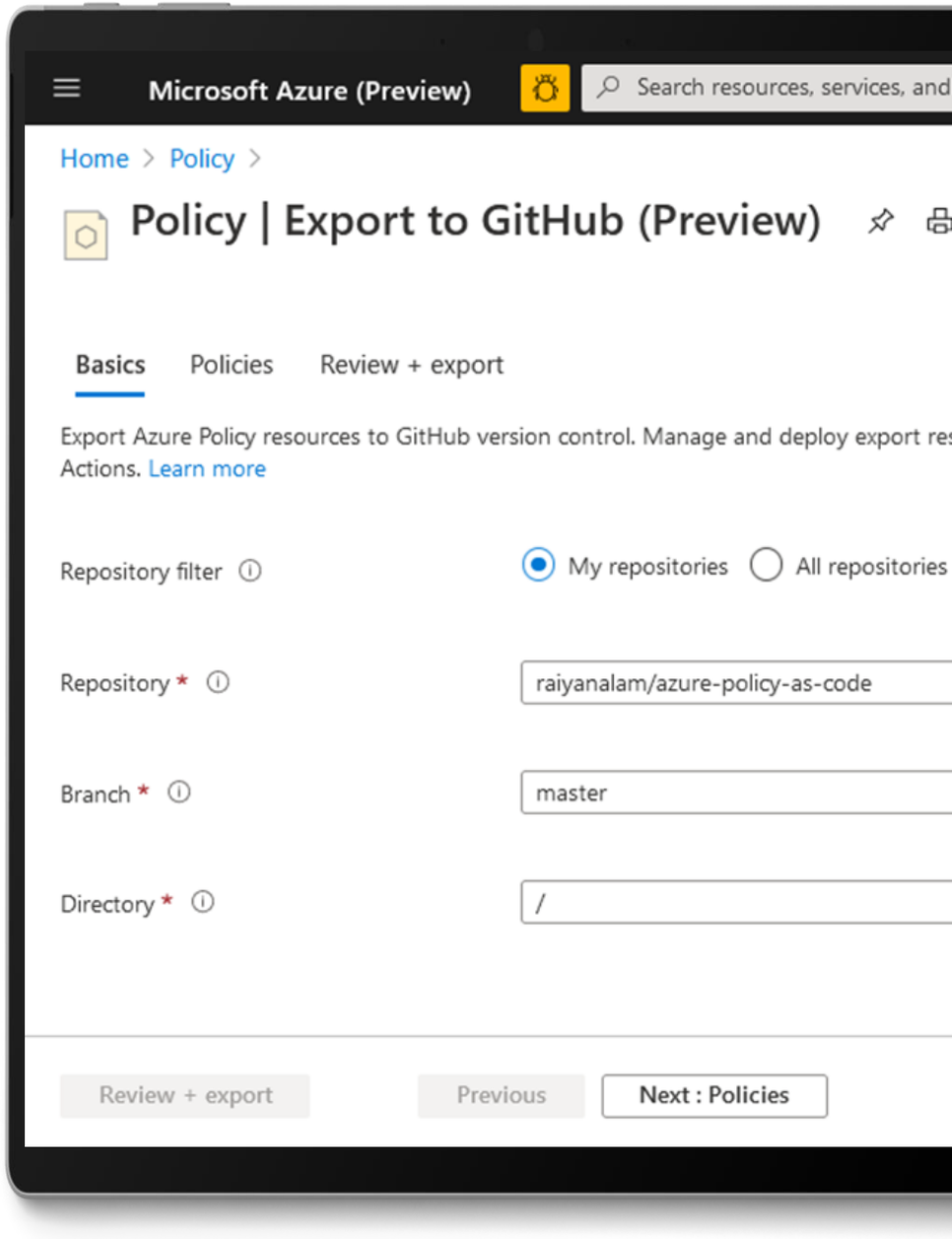
Easily manage Azure Policies "as code" from a GitHub repository in an orchestrated manner

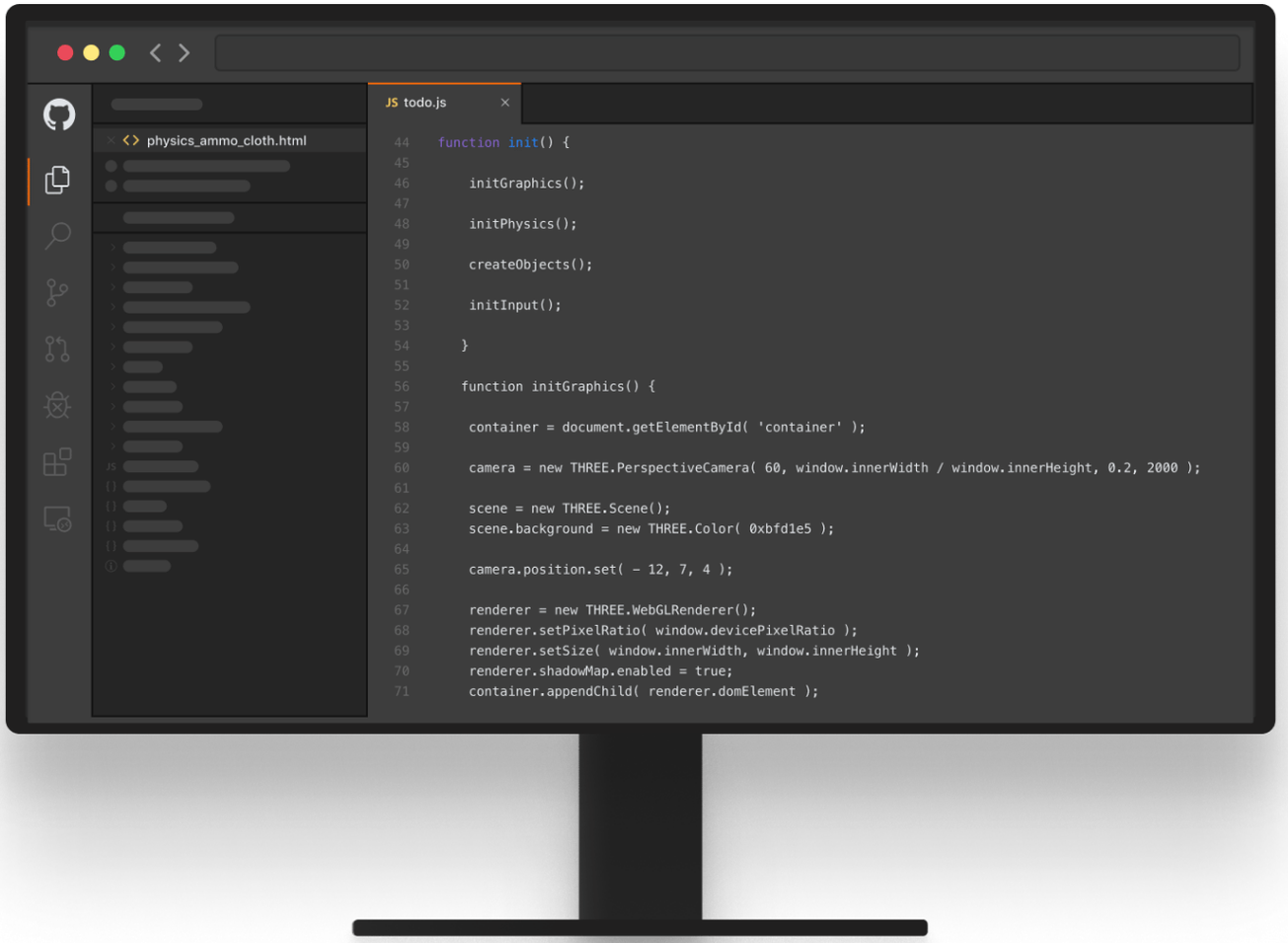
Scan containers

Scan for common vulnerabilities in Docker images before pushing them to a container registry or deploying them to a containerized web app or Kubernetes cluster

Manage secrets using Azure Key Vault

Dynamically pull secrets from an Azure Key Vault instance for consumption in GitHub Action workflows





SECURING THE DEVELOPER WORKFLOW

GitHub Advanced Security

A DEVELOPER-FIRST, COMMUNITY DRIVEN APPROACH

DEPENDENCY SCANNING

- Alerts and security updates for new vulnerabilities
- Integrated review when introducing new dependencies

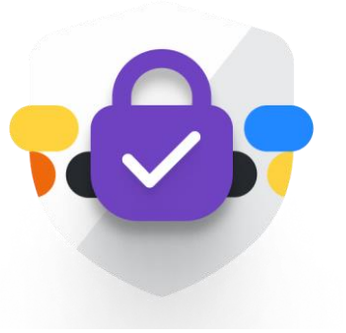
CODE SCANNING

- Extensible framework for code scanning
- Integrated within the developer workflow
- Backed by industry-leading CodeQL engine

SECRET SCANNING

- Scanning for leaked secrets in public and private repos
- Write custom patterns or rely on 100+ default ones

Secure your complete software lifecycle with GitHub



Dependency updates

Identify vulnerable dependencies, and update them to secure versions automatically

Code scanning

Scan your code for vulnerabilities, powered by CodeQL, and the work of the security research community

Secret scanning

Prevent secrets from leaking into code, including automatic remediation with 35+ providers

Included w/ GitHub Enterprise

GitHub Advanced Security (GHAS)

Secure your complete software lifecycle with Azure DevOps

Integrated Security Analysis
with Team Build

ARM Template Checker
Security Verification Tests

Security Vulnerabilities
Ensure license compliance

Code Quality Gate

Azure DevOps Marketplace

Defender for DevOps Preview

Defender for DevOps fills five vital needs for managing the security of code and code management systems:

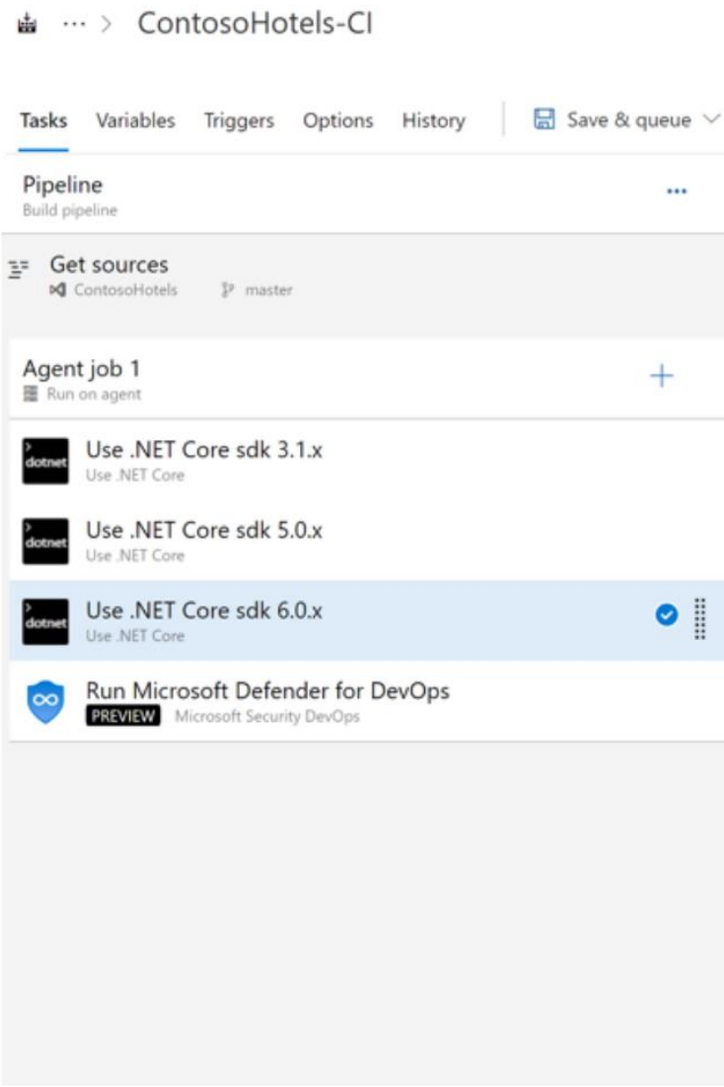
Vulnerabilities in Code
Keep dependencies up-to-date with automated pull requests
Detect and monitor for leaked credentials and secrets

Secure and Compliant Infrastructure-as-Code (IaC)
Deploy and enforce policy to ensure uniformity and best practices
Find and fix issues before they are deployed, prevent drift

Security Monitoring
Respond to suspicious activities in code, pipelines, and the developer cloud
Assess the impact of vulnerabilities and risk clearly and easily

Continuous Cloud Security and Compliance
Assess and view state of pre-production resources
Compare posture to security and compliance standards
Leverage attack graphs & attack simulation

Secure Cloud-Native Workloads
Multi-cloud integration, Containers, Serverless, APIs



Encrypt data

- In transit
- At rest
- While it’s being processed

Confidential computing allows you to isolate your sensitive data while it's being processed. Use confidential computing to protect data by using confidential computing to:

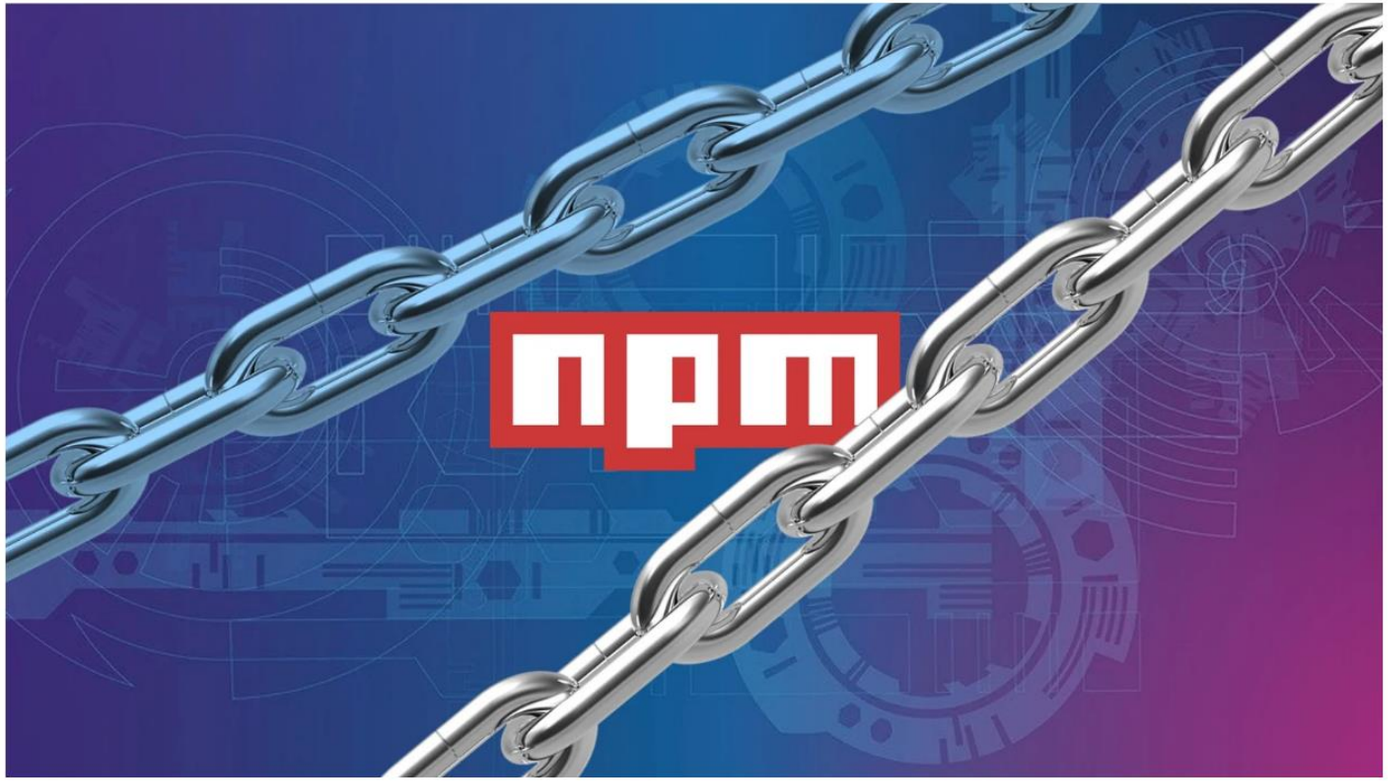
•Secure financial data
•Protect patient information
•Run machine learning processes on sensitive information
•Perform algorithms on encrypted data sets from multiple sources

January, 2022

Dev corrupts NPM libs 'colors' and 'faker' breaking thousands of apps

By Ax Sharma

January 9, 2022 09:17 AM 32



Users of popular open-source libraries 'colors' and 'faker' were left stunned after they saw their applications, using these libraries, printing gibberish data and breaking.

Some surmised if the NPM libraries had been compromised, but it turns out there's much more to the story.

The benefits of DevSecOps

MORE SECURE CODE, SHIPPED AT THE SAME SPEED

Reduce remediation time by shifting security left

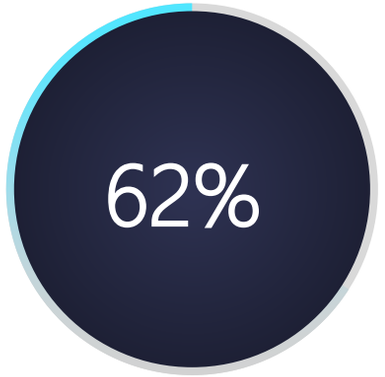
Integrate with and secure your existing toolchains

Quickly identify new threat vectors

Importance of shifting security left

reduction in security incidents by extending security to development[2]

Security cost to fix a security defect in production versus in development[1]



of enterprises do not integrate security in the development phase[3]

[1]https://www.gartner.com/doc/reprints?id=1-265CMWW4&ct=210527&st=sb
[2]https://www.gartner.com/smarterwithgartner/is-the-cloud-secure/
[3]Sources: McKinsey Developer Velocity, Microsoft Enterprise DevOps Report, GitHub Octoverse Report 2020

Barriers to DevSecOps adoption

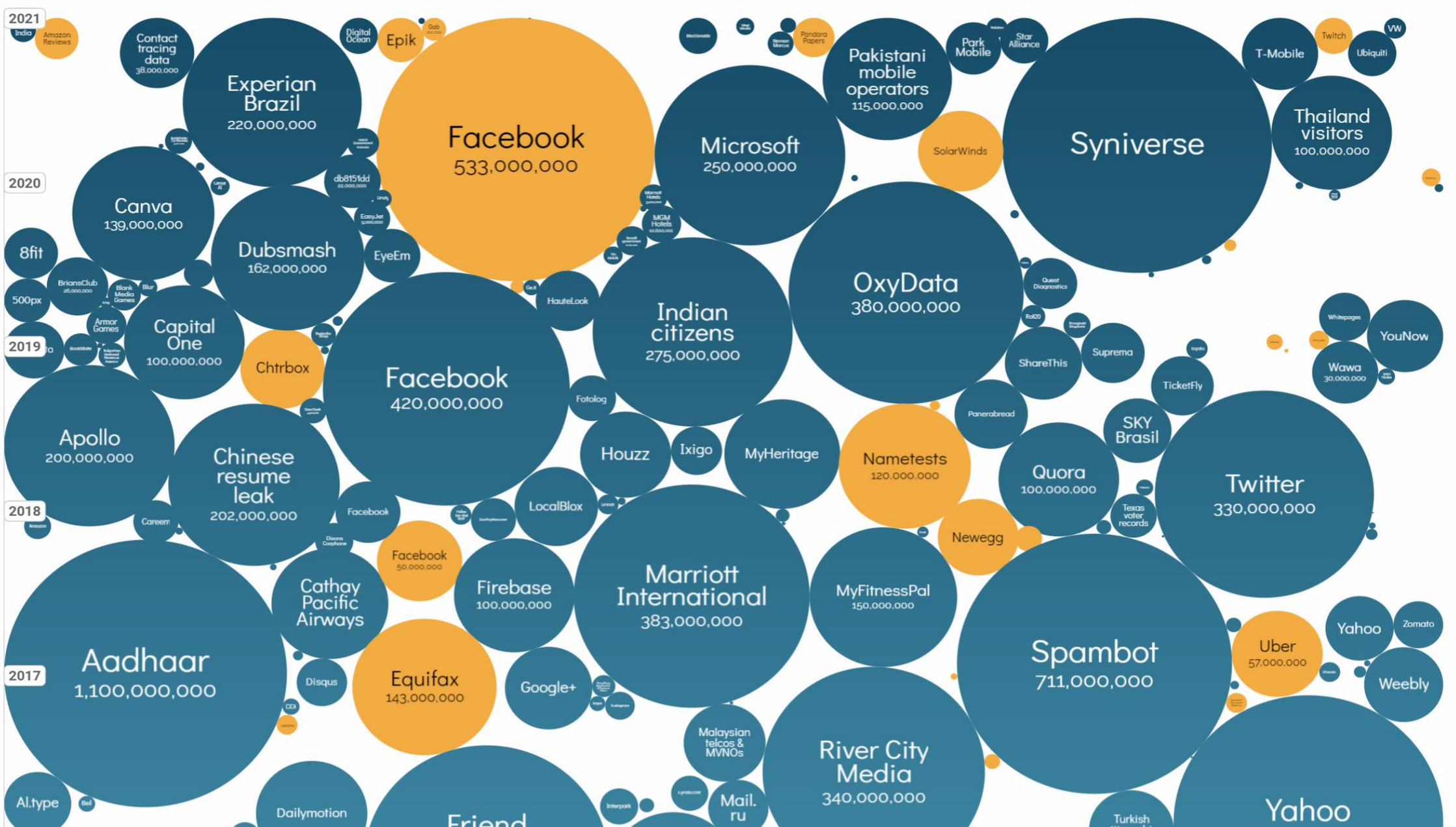
WHY IS DEVSECOPS HARDER TO ADOPT THAN DEVOPS

Organization and team gaps

Skill and knowledge gaps

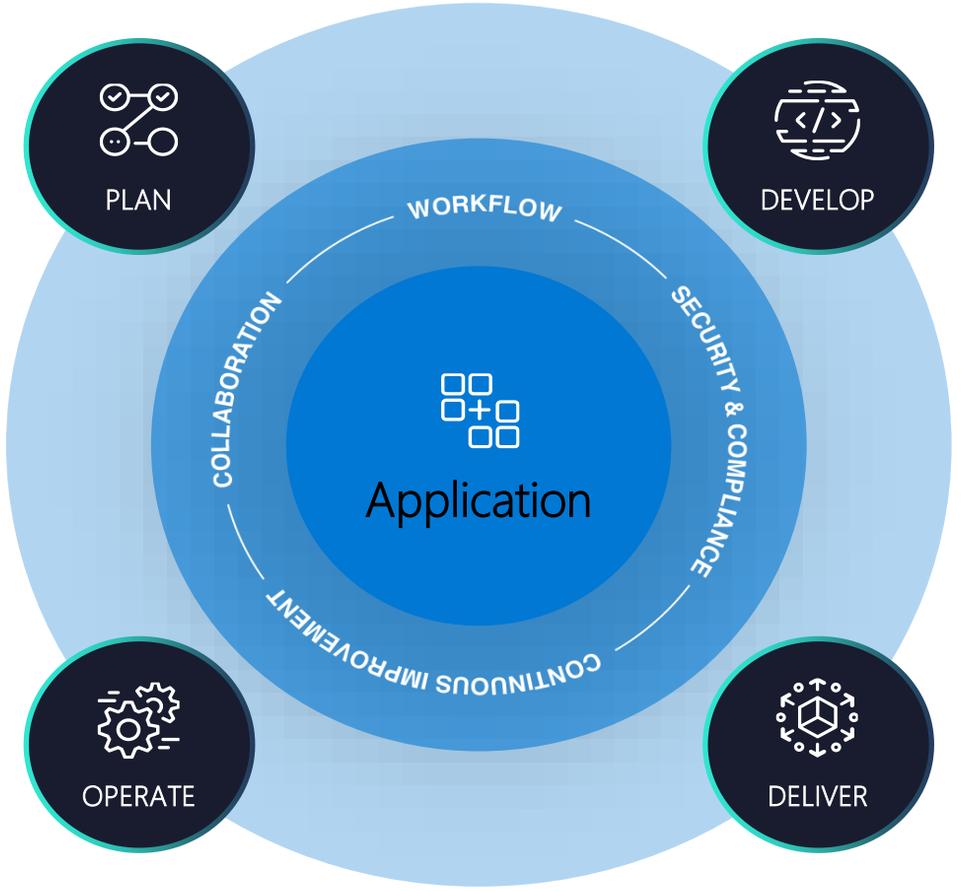
Solutions aren't built for developers

World's Largest Data Breaches & Hacks



information is beautiful

Application security in a wider security context



SDL Definition

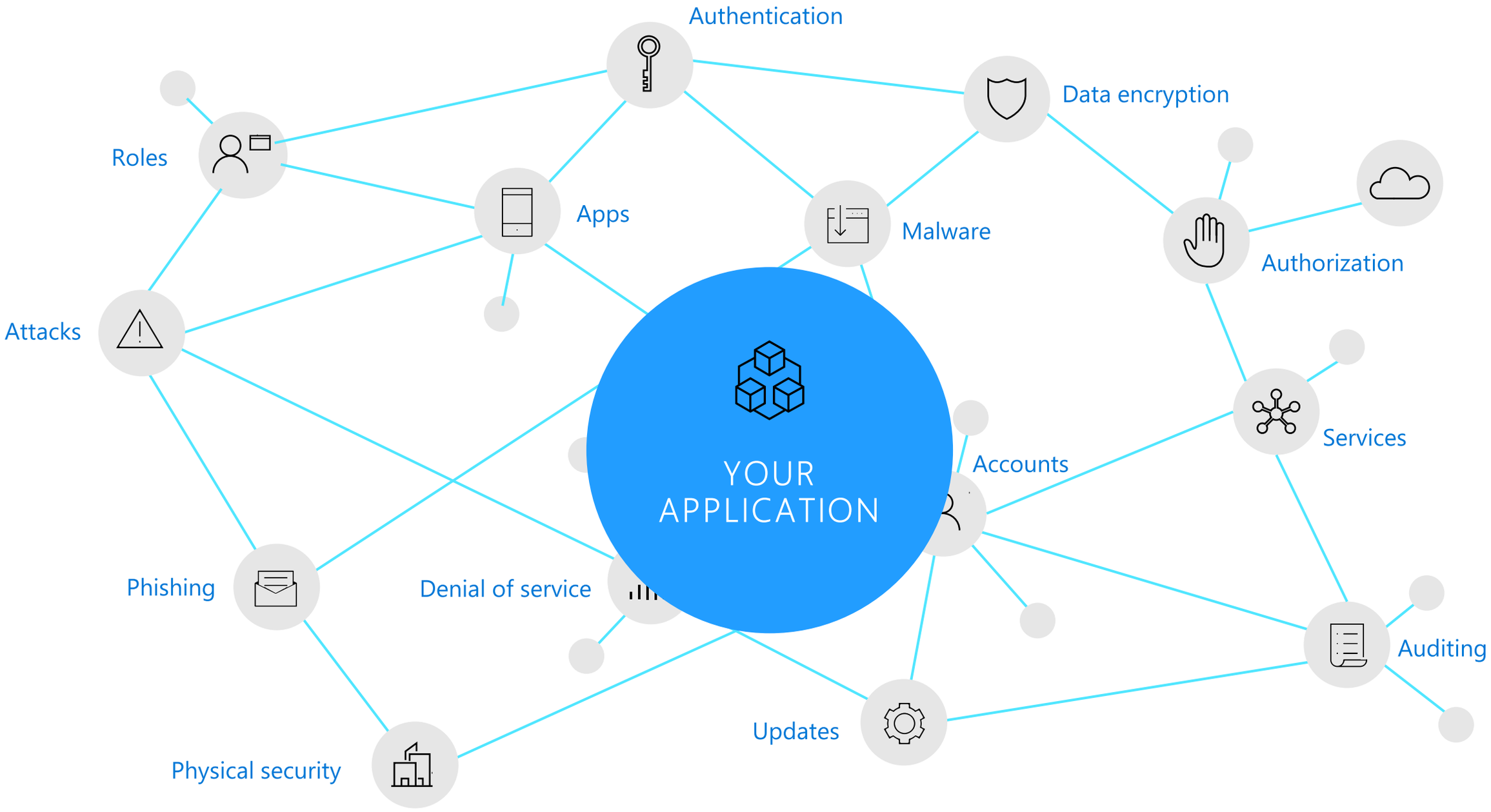
The Security Development Lifecycle (SDL) consists of a set of practices that support security assurance and compliance requirements. The SDL helps developers build more secure software by reducing the number and severity of vulnerabilities in software.

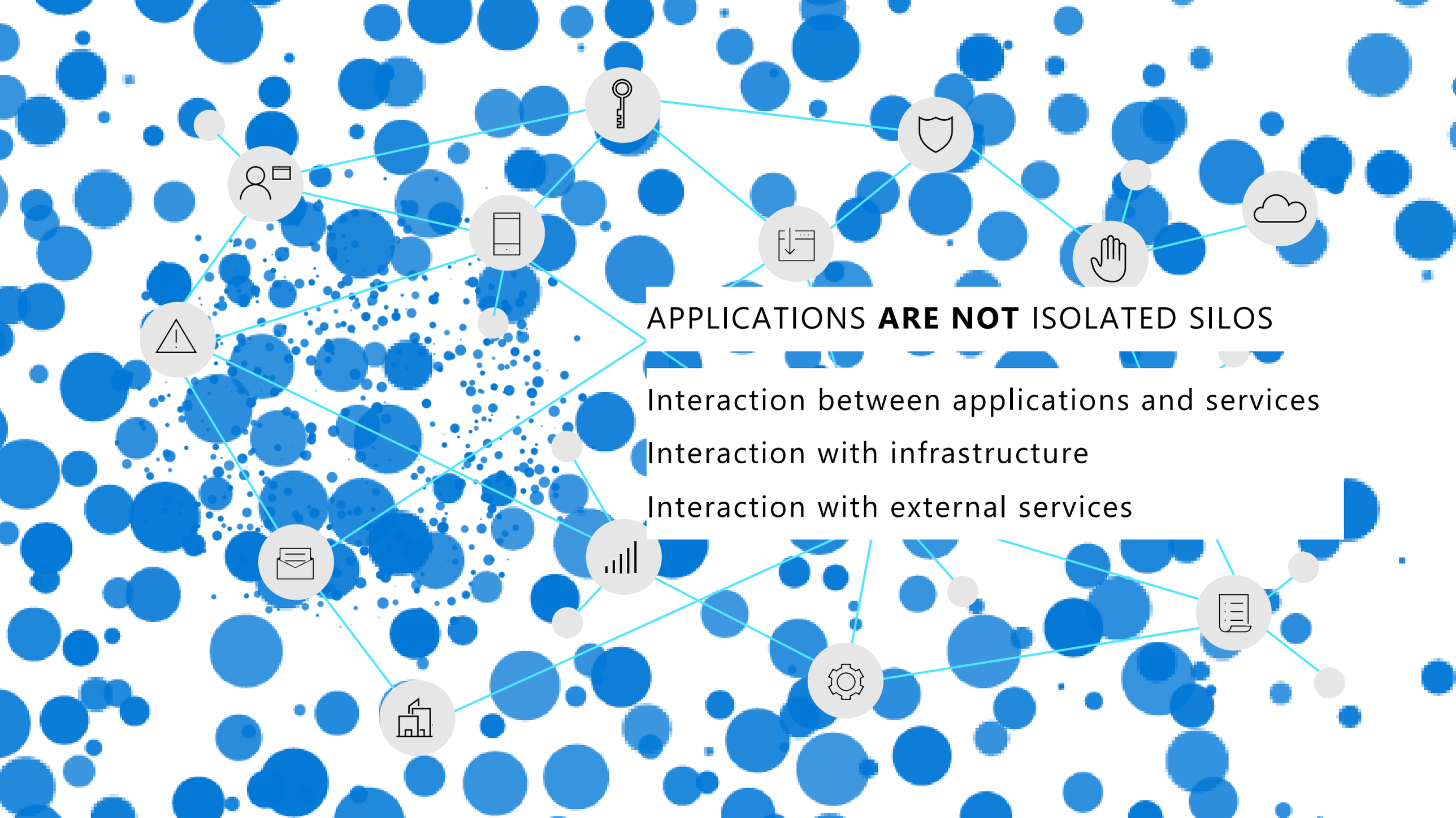
DevSecOps Definition (Development + Security + Operations)

DevSecOps is an evolution in the way development organizations approach security by introducing a security-first mindset culture, and automating security into every phase of the software development lifecycle from design to delivery.

AppSec Definition

AppSec is the process of finding, fixing, and preventing security vulnerabilities at the application level.


Authentication
Data encryption
Roles
Apps
Malware
Authorization
Attacks
YOUR APPLICATION
Accounts
Services
Phishing
Denial of service
Auditing
Updates
Physical security



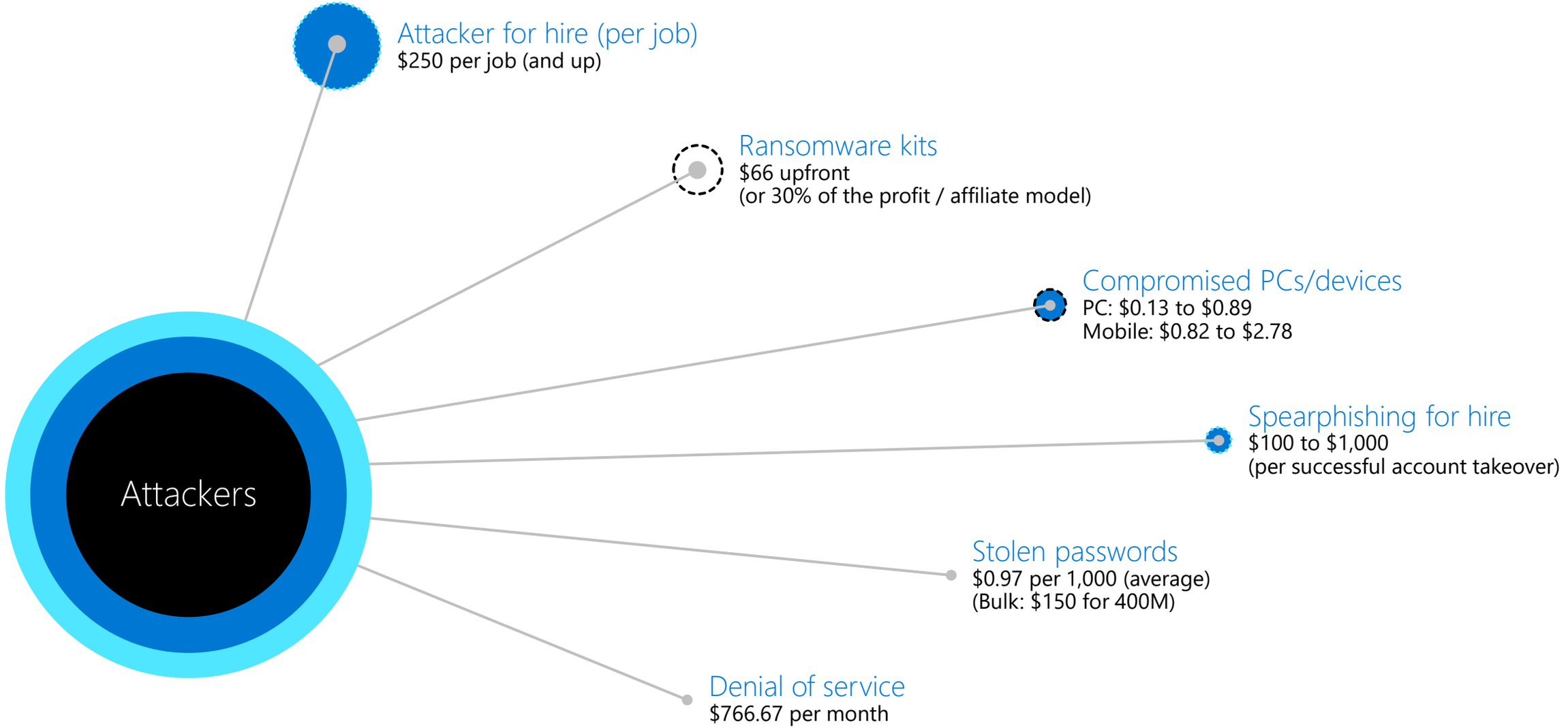
APPLICATIONS **ARE NOT** ISOLATED SILOS

Interaction between applications and services

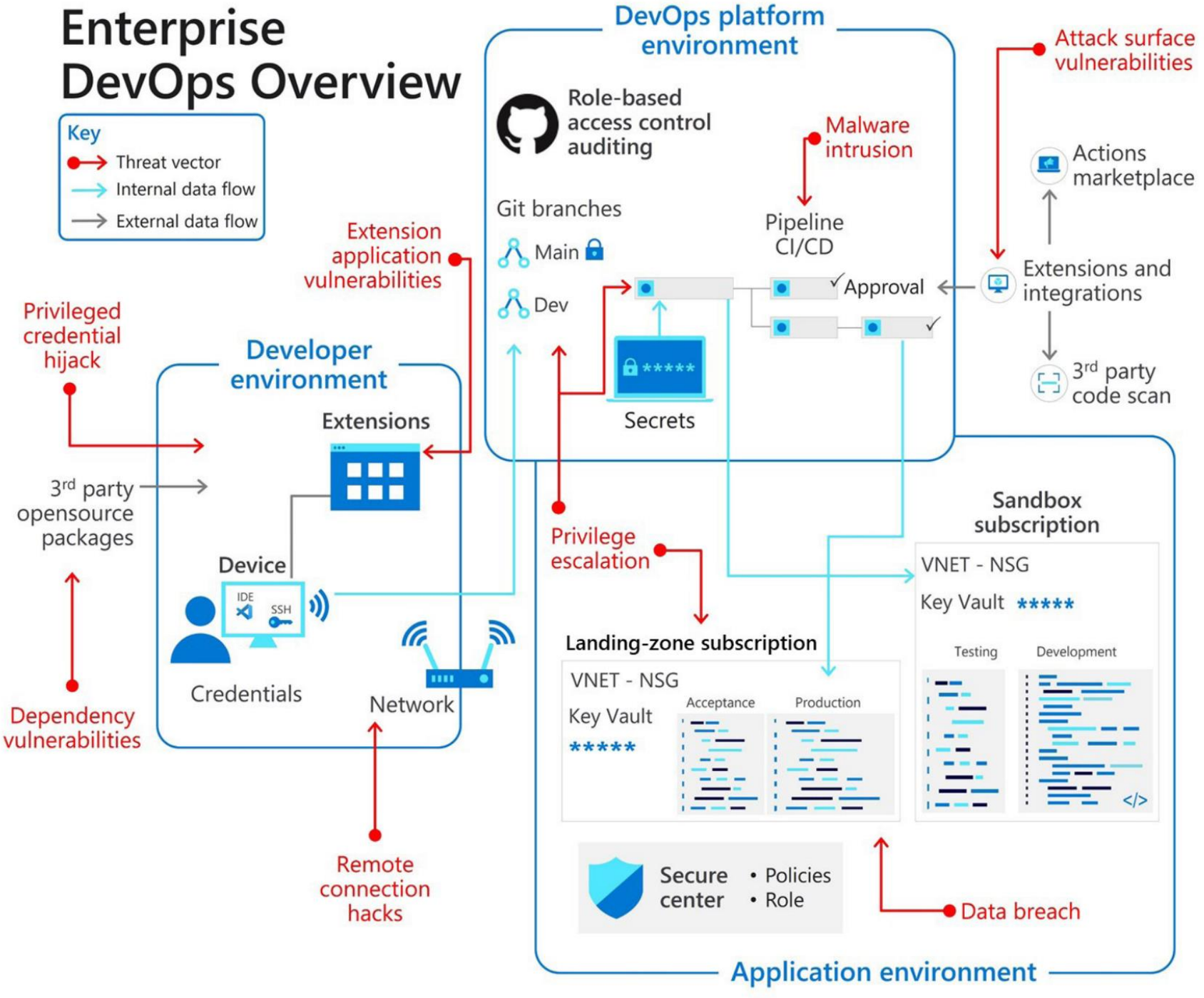
Interaction with infrastructure

Interaction with external services

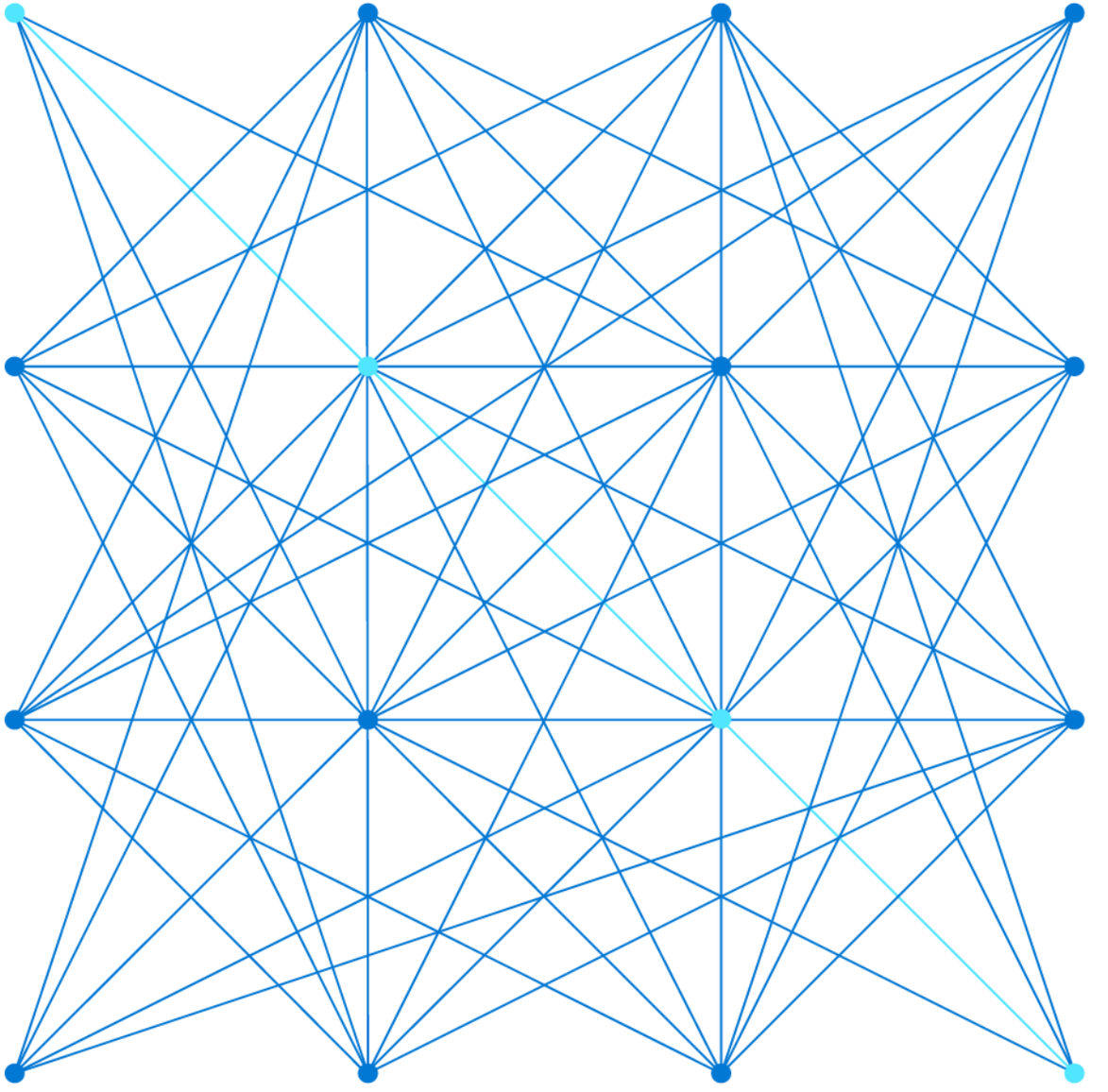
For sale in “bad neighborhoods” on the internet



Enterprise DevOps

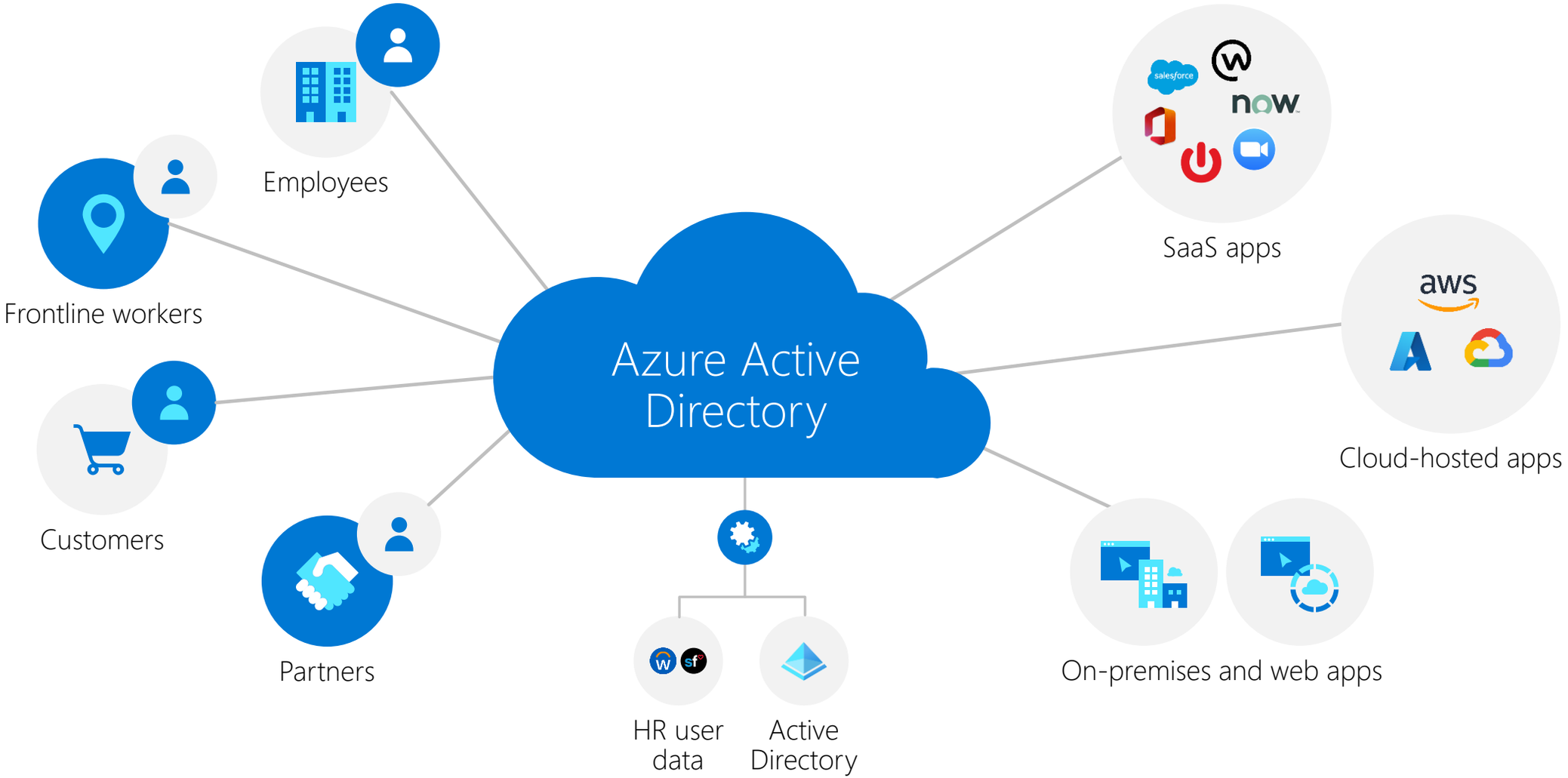


Azure Security



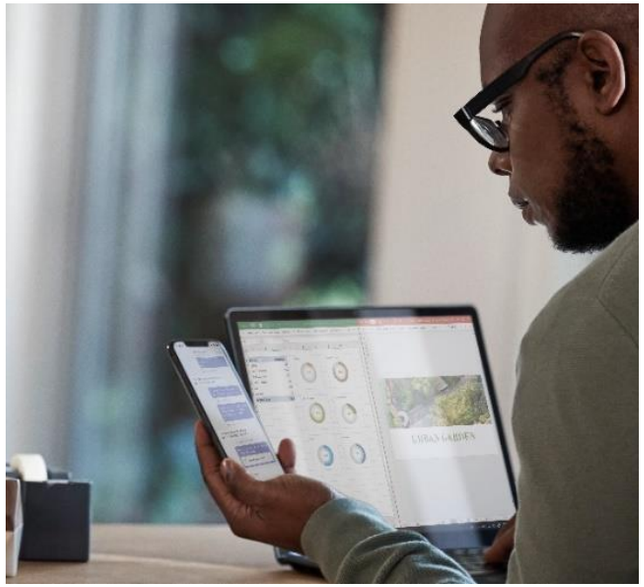
Unified identity management

Manage all your identities and access to all your applications across your hybrid environment



Multi-factor authentication

Verify user identities with strong authentication

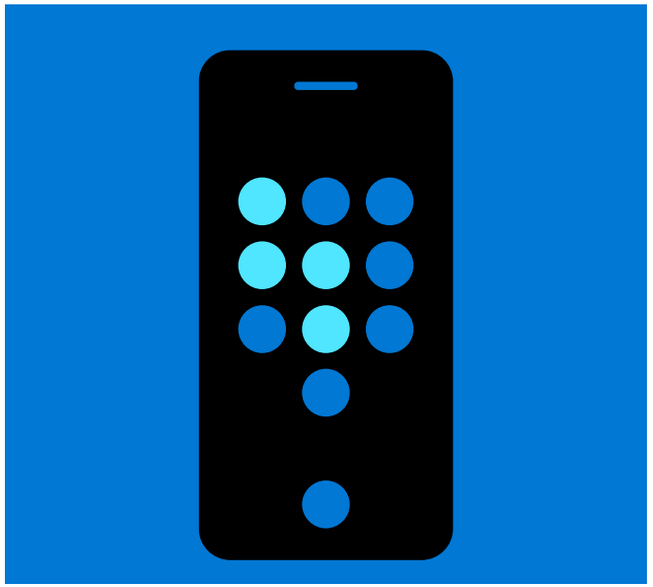


We support a broad range of multi-factor authentication options

Including passwordless technology

- Microsoft Authenticator
- Windows Hello
- FIDO2 Security key
- Biometrics

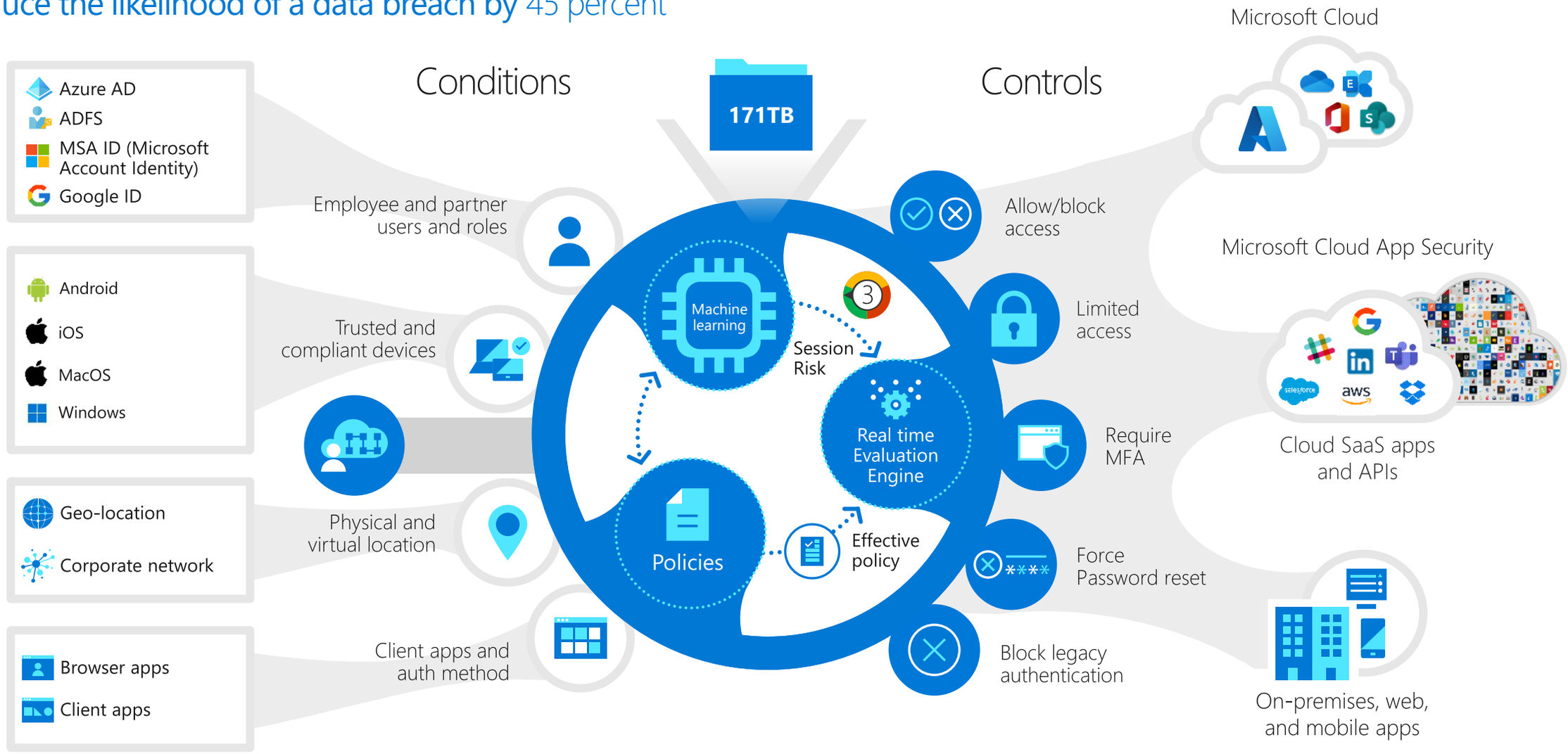
- Push notification
- Soft tokens OTP
- Hard tokens OTP
- SMS, Voice



Multi-factor authentication prevents 99.9% of identity attacks

Provide just-in-time and just-enough access

Reduce the likelihood of a data breach by 45 percent

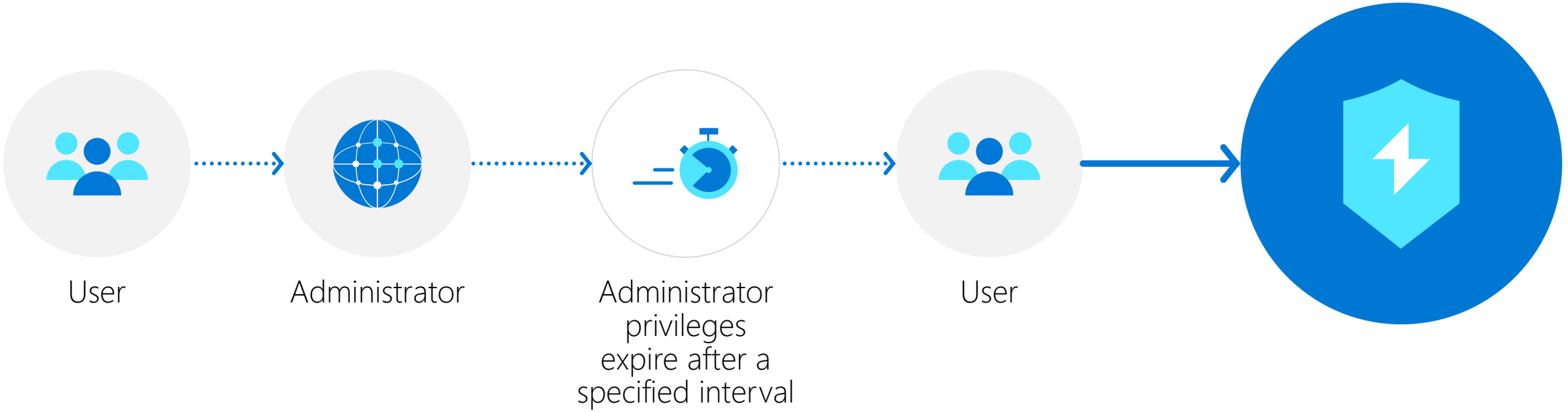


Source: Forrester Consulting, "The Total Economic Impact™ of Securing Apps with Microsoft Azure Active Directory," August 2020, commissioned by Microsoft.

Privileged identity management

Discover, restrict, and monitor privileged identity access

- Enforce on-demand, just-in-time administrative access when needed
- Ensure policies are met with alerts, audit reports and access reviews
- Manage admins access in Azure AD and also in Azure RBAC



Role-based access controls

Extend fine-grained access management to cloud resources

Principal

User

Group

Service principal

Managed identity

Role

Built In

Owner

Contributor

Reader

...

Security Reader

Custom

Support Ticket Reader

Scope

Management Group

Subscription

Resource Group

Virtual Machines

Database

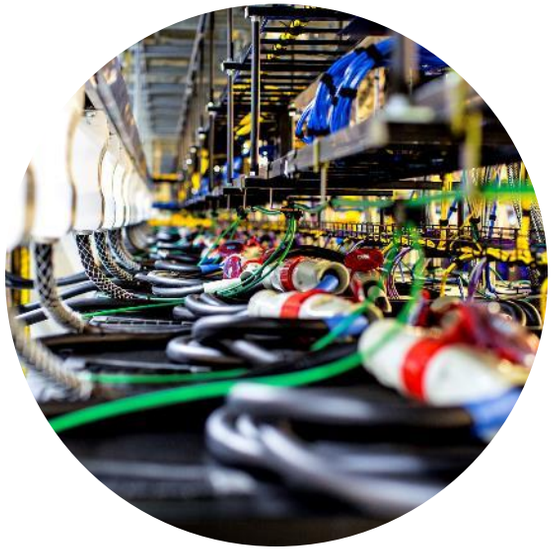
The era of flux and transformation

Adaptable attackers

Attacks traverse laterally across silos and perimeters

Disparate security tools

Security tools are increasingly complex, and poorly integrated into the DevOps cycle



Overwhelming noise

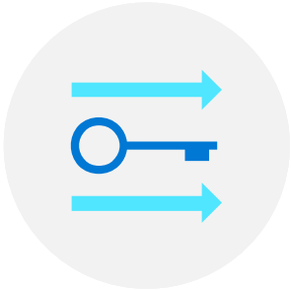
It's harder than ever to find the signal in the noise

The result? An empowered team

Technology

Azure built-in controls

Identity and access management



Security posture management

Threat protection

Apps & data security

Network security

Defense in depth

Technology

Defense in depth

Identity & access	Security management	Threat protection	Apps security	Data security	Network security
Role based access	Cloud security posture management	SIEM	Dependency management	Encryption	DDoS protection
Multi-factor authentication	Policy and governance	Extended Detection and Response (XDR)	Code security	Confidential computing	Firewall
Central identity management	Regulatory compliance	External Attack Surface Management (EASM)	Key management	Auditing	Web app firewall
Identity protection	Unified security management	VM & cloud native asset protection	Certificate management	Data masking & erasure	Private connections
Privileged identity management	Hybrid and multi-cloud	Antimalware		Information protection & governance	NetSec policy management
		IoT Security			

Microsoft + Partners

Azure security technology and services

Protect Azure and your entire multi-cloud environment with built-in security, powered by AI

Identity and access management

Azure Active Directory
Your universal platform to manage and secure all your users and data

Security posture management

Microsoft Defender for Cloud
Security posture management for your multicloud environments

Threat protection

Microsoft Sentinel
Intelligent security analytics for your entire enterprise

RiskIQ EASM
Discover and monitor for your global attack surface

RiskIQ threat intelligence
Adversary-threat insights relevant to your external attack surface

Microsoft Defender for Cloud
Built-in threat protection for multicloud and hybrid workloads

Microsoft Defender for IoT
Agentless asset discovery, vulnerability management, and threat detection for IoT/OT devices

Apps & data security

Azure confidential computing
Protect your data and code while in use in the cloud

Azure Key Vault
Safeguard cryptographic keys and other secrets

Azure attestation
Store and process confidential data with confidence

Azure dedicated HSM
Your hardware security module (HSM) in the cloud

GitHub
Build secure apps rights from the start

Network security

Azure Firewall
Cloud-native firewall to protect Azure virtual networks

Azure Firewall Manager
Central network security policy management

Azure Web Application Firewall
Protect web/mobile apps and APIs from common web vulnerabilities

Azure Front Door
Fast, reliable and secure cloud CDN with threat protection

Azure DDoS protection
Always-on monitoring to protect against DDoS attacks

Azure Bastion
Private and fully managed RDP and SSH access to your VMs

Key actions for all Azure customers:

Turn on secure score

Gain insight into the security state of your cloud workloads

Turn on Microsoft Defender for all cloud workloads

Protect your workloads with built-in threat protection

Turn on WAF and DDoS Protection for every website

Protect your web applications from malicious attacks

Turn on Azure Firewall for every subscription

Protect your Azure virtual network resources


Microsoft