Secure your web applications on Azure and protect your apps against the most common and dangerous web application attacks.

Learning objectives

In this module, you will:

- Use Azure Security Center
- Verify your application's inputs and outputs
- Store your secrets into Key Vault
- Ensure you are using the latest version of your framework, and its security features
- Validate that your program dependencies and libraries are safe to use

Start

Bookmark

Add to collection

Prerequisites

None

This module is part of these learning paths

- Secure your cloud applications in Azure
- Secure your cloud data

- Introduction 2 min
- Azure Security Center 15 min
- Inputs and Outputs 10 min
- Secrets in Key Vault 5 min
- Framework Updates 7 min
- Safe Dependencies 5 min
- Conclusion 1 min

Introduction

- 2 minutes

Security is top-of-mind for most people working in technology, and this is especially true when it comes to software. It seems we are constantly hearing about companies and governments that we entrusted our private data to, being compromised. When this happens - either maliciously or accidentally, it costs them customers and, ultimately money.

Did you know that the most common cause of data breaches is poor security in software? It's true. This means that the pressure is on for software developers to be more diligent than ever. But where do they begin? This module is the start of your journey into the world of application security, with the top five defenses for web applications. Learn how to secure your web applications on Azure and protect your apps against the most common and dangerous web application attacks.

Learning objectives

In this module, you will:

- Use Azure Security Center
- Verify your application's inputs and outputs
- Store your secrets into Key Vault
- Ensure you are using the latest version of your framework, and its security features
- Validate that your program dependencies and libraries are safe to use

简介

- 2 分钟

对于大多数从事技术工作的人来说，安全性是最重要的，尤其是在软件方面。 我们似乎不断听到有消息称，受托处理我们的私人数据的公司和政府受到安全威胁。 当这种情况发生时 - 无论是恶意的还是意外的，他们都要付出客户的代价，最终还要付出金钱。
你知道造成数据泄露的最常见原因是软件的安全性差吗？ 这是真的。 这意味着，软件开发人员备受压力，不得不比以往更加努力。 但他们从哪里开始呢？ 本模块引领你进入应用程序安全领域的开始，其中包含 Web 应用的前五大防御措施。 学习如何在 Azure 上保护 Web 应用，并保护应用免受最常见和最危险的 Web 应用攻击。

学习目标

在本模块中，将执行以下操作：

- 使用 Azure 安全中心
- 验证应用程序的输入和输出
- 将机密存储到密钥保管库
- 确保你使用的是最新版本的框架，并使用其安全功能
- 验证程序依赖项和库是否可以安全使用

下一单元: Azure 安全中心

继续

Azure 安全中心

- 15 分钟

确定所有需要保护的区域以及先于黑客一步找到漏洞，是实现安全性的最大难题之一。Azure 提供一个称为 Azure 安全中心的服务来大幅简化此工作。

什么是 Azure 安全中心?

Azure 安全中心 (ASC) 是一个监视服务，针对 Azure 中和本地的所有服务提供威胁防护。它可以：

- 根据配置、资源和网络提供安全建议。
- 监视本地和云工作负荷的安全设置，并自动将所需安全性应用到联机的新服务。
- 持续监视所有服务并执行自动安全评估，以识别潜在漏洞，避免这些漏洞被恶意利用。
- 使用机器学习来检测恶意软件，并阻止在服务与虚拟机中安装这些软件。 还可以将应用程序加入允许列表，确保只允许执行经过验证的应用。
- 分析和识别潜在的入站攻击，帮助调查可能发生的威胁和任何后期违规活动。
- 端口的实时访问控制，确保网络只允许所需的流量，从而减小受攻击面。

ASC 属于 Internet 安全中心 (CIS) 建议的服务。

激活 Azure 安全中心

Azure 安全中心可为混合云工作负载提供统一的安全管理和高级威胁防护，并且它分为两个层级：免费层和标准层。 免费层提供安全策略、评估和建议，而标准层则提供一组强大的功能（包括威胁智能）。

鉴于 ASC 的优势，你公司的安全团队决定为办公室的所有订阅启用 ASC。 你在今天早上收到一封电子邮件，其中要求为应用程序启用 ASC。让我们了解如何执行此操作。

重要

免费版 Azure 沙盒不支持 Azure 安全中心。 你可在自己的订阅中执行下列步骤，也可按照步骤了解如何激活 ASC。

1. 打开 Azure 门户 ，并在左侧菜单中选择“Azure 安全中心”；如果看不到该选项，可按如下所示，在“安全性”部分选择“所有服务”并找到“安全中心”。

2. 如果你之前从未打开过 ASC，窗格将首先显示“入门”项，其中可能要求你升级订阅。 暂时请忽略该消息，选择页面底部的“跳过”，然后选择“概述”。
    - 此时会显示订阅中所有可用元素的“总体安全态势”。

- 其中包含大量可供浏览的有用信息。

3. 接下来，在“策略与符合性”下选择“涵盖范围”。 这会显示 ASC 涵盖（或未涵盖）的订阅元素。 在此处，可以为自己有权访问的任何订阅启用 ASC。 尝试切换以下三个覆盖率区域：“未覆盖”、“基本覆盖率”和“标准覆盖率”。

4. 未覆盖的订阅会显示 ASC 激活提示。 可按下“立即升级”按钮，为订阅中的所有资源启用 ASC。

免费与标准定价层

尽管可将免费的 Azure 订阅层与 ASC 配合使用，但这样做仅限于评估 Azure 资源和提供其建议。 若要真正利用 ASC，需要如上所示升级到标准层订阅。 可以通过“涵盖范围”窗格中的“立即升级”按钮（如前所述）升级订阅。 还可以在 ASC 菜单中切换到“入门”窗格，其中会逐步引导你更改订阅级别。 定价和功能可能根据区域的不同而不同，可在定价页中获取完整概述。

备注

若要将订阅升级到标准层，必须拥有订阅所有者、订阅参与者或安全管理员角色。

重要

30 天试用期结束后，ASC 标准版将以每月每个节点 15 美元的价格对你的帐户进行计费。

禁用 Azure 安全中心

对于生产系统，肯定需要将 Azure 安全中心保持启用状态，以便可以监视所有资源是否受到威胁。 但是，如果你只是出于体验的目的启用了 ASC，则最好是将其禁用，以免产生费用。 现在就让我们执行此操作。

1. 打开 Azure 门户 ，并在左侧菜单中选择“Azure 安全中心”；如果看不到该选项，可按如下所示，在“安全性”部分选择“所有服务”并找到“安全中心”。

2. 从左侧菜单中选择“安全策略”。

3. 接下来，选择要为其降级 ASC 的订阅旁边的“编辑设置 >”。

4. 在下一个屏幕上，从左侧菜单中选择“定价层”。

5. 此时会显示一个如下图所示的新页面。 单击左侧显示“免费(仅限 Azure 资源)”的框。

6. 按下屏幕顶部的“保存”按钮。

现已将订阅降级到 Azure 安全中心的免费层。
祝贺你！你现已采取第一项（也是最重要的一项）措施来保护应用程序、数据和网络！

知识检查

1. 判断正误：Azure 安全中心适用于 Azure 资源和本地资源。

正确

错误
检查你的答案

A
z
u
r
e
S
e
c
u
r
it

y C e n t e r

- 15 minutes

One of the biggest problems with security is being able to see all the areas you need to protect and to find vulnerabilities before hackers do. Azure provides a service which makes this much easier called Azure Security Center.

What is Azure Security Center?

Azure Security Center (ASC) is a monitoring service that provides threat protection across all of your services both in Azure, and on-premises. It can:

- Provide security recommendations based on your configurations, resources, and networks.
- Monitor security settings across on-premises and cloud workloads and automatically apply required security to new services as they come online.
- Continuously monitor all your services and perform automatic security assessments to identify potential vulnerabilities before they can be exploited.
- Use machine learning to detect and block malware from being installed in your services and virtual machines. You can also allowlist applications to ensure that only the apps you validate are allowed to execute.
- Analyze and identify potential inbound attacks and help to investigate threats and any post-breach activity which might have occurred.
- Just-In-Time access control for ports, reducing your attack surface by ensuring the network only allows traffic you require.

ASC is part of the Center for Internet Security (CIS) recommendations.

Activating Azure Security Center

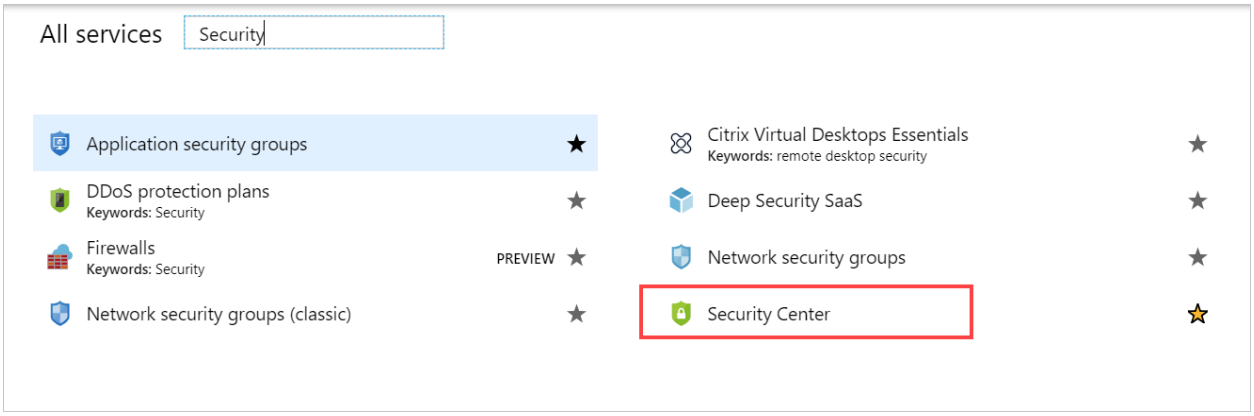
Azure Security Center provides unified security management and advanced threat protection for hybrid cloud workloads and is offered in two tiers: Free and Standard. The free tier provides

security policies, assessments, and recommendations while the Standard tier provides a robust set of features, including threat intelligence.
Given the benefits of ASC, the security team at your company has decided that it be turned on for all subscriptions at your office. You got an email this morning to turn it on for your applications - so let's look at how to do that.

Important

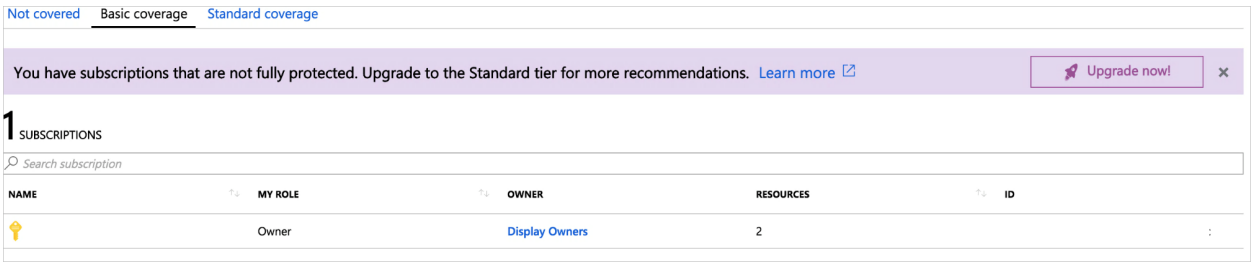
Azure Security Center is not supported in the free Azure sandbox. You can perform these steps in your own subscription, or just follow along to understand how to activate ASC.

1. Open the Azure portal and select **Azure Security Center** from the left-hand menu, if you don't see it there, you can select **All services** and find **Security Center** in the security section as shown below.



2. If you have never opened ASC, the pane will start on the **Getting started** entry which might ask you to upgrade your subscription. Ignore that for now, select **Skip** at the bottom of the page, and then select **Overview**.
    - This will display the "big security picture" across all the elements available in your subscription.
    - This has a ton of great information you can explore.
3. Next, select **Coverage**, under "Policy and Compliance". This will display what subscription elements are being covered (or not covered) by ASC. Here you can turn on ASC for any subscription you have access to. Try switching between the three coverage areas: "Not covered", "Basic coverage" and "Standard coverage".

4. Subscriptions that are not covered will have a prompt to activate ASC. You can press the "Upgrade Now" button to enable ASC for all the resources in the subscription.



Free vs. Standard pricing tier

While you can use a free Azure subscription tier with ASC, it is limited to assessments and recommendations of Azure resources only. To really leverage ASC, you will need to upgrade to a Standard tier subscription as shown above. You can upgrade your subscription through the "Upgrade Now" button in the **Coverage** pane as noted above. You can also switch to the **Getting Started** pane in the ASC menu which will walk you through changing your subscription level. The pricing and features may change based on the region, you can get a full overview on the pricing page.

Note

To upgrade a subscription to the Standard tier, you must be assigned the role of Subscription Owner, Subscription Contributor, or Security Admin.

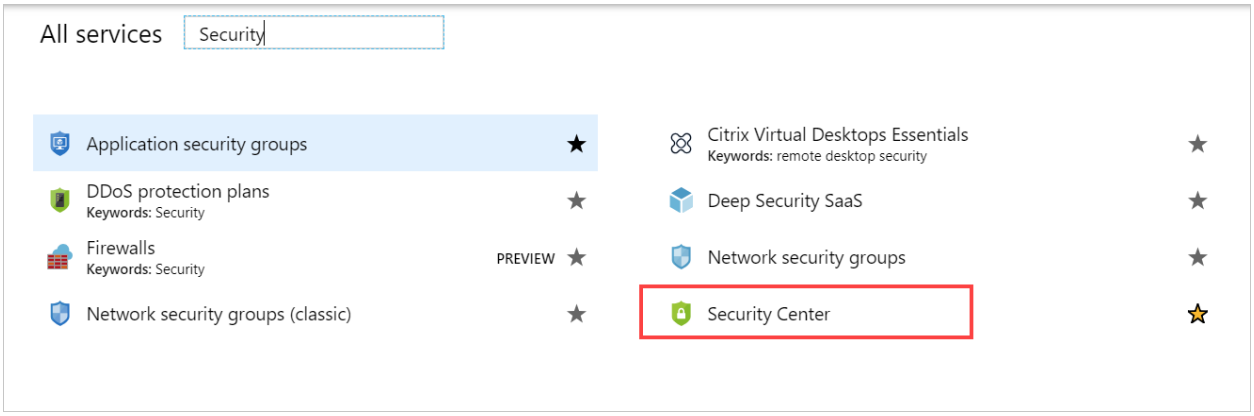
Important

After the 30-day trial period is over, ASC Standard is priced at **$15/node per month** and will be billed to your account.

Turning off Azure Security Center

For production systems, you will definitely want to keep Azure Security Center turned on so it can monitor all your resources for threats. However, if you are just playing with ASC and turned it on, you will likely want to disable it to ensure you are not charged. Let's do that now.

1. Open the Azure portal and select **Azure Security Center** from the left-hand menu, if you don't see it there, you can select **All services** and find **Security Center** in the security section as shown below.

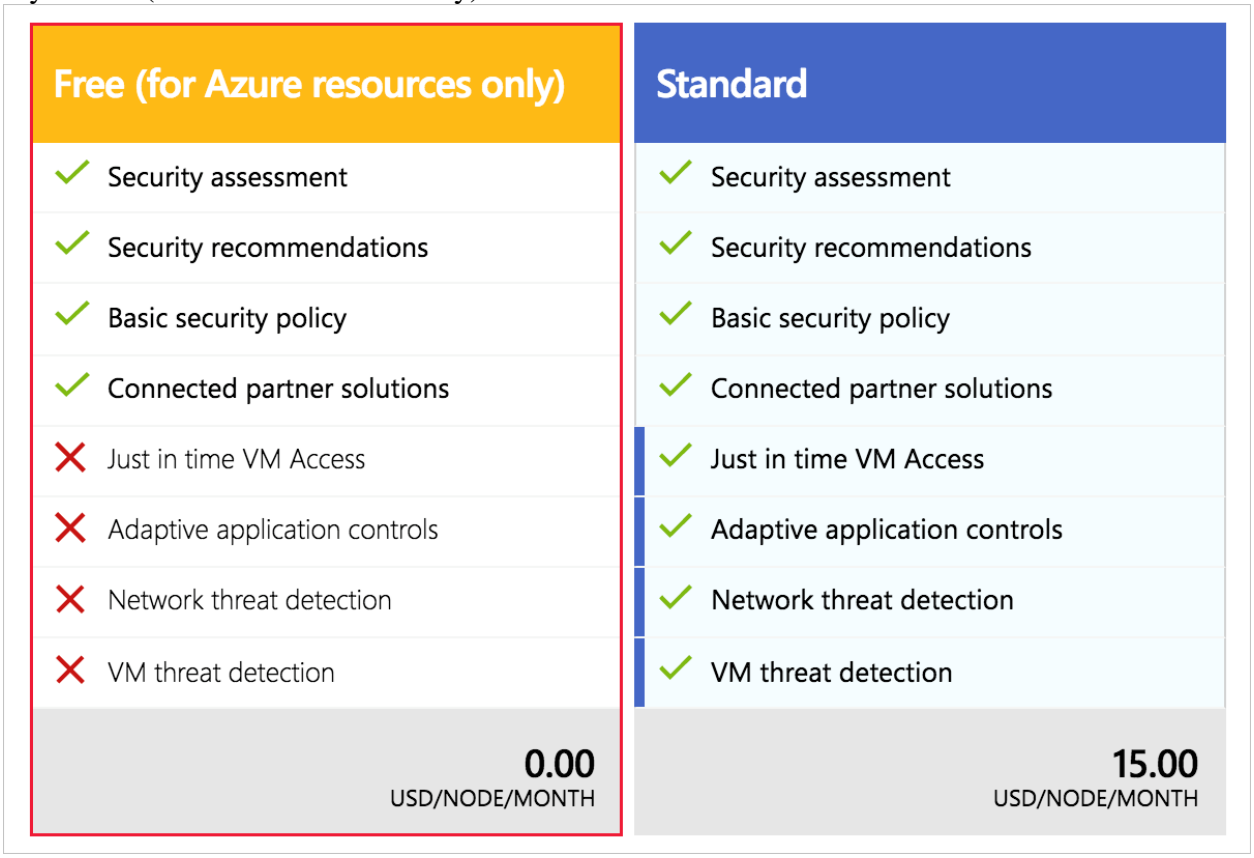


2. Select **Security Policy** from the left-hand menu.

3. Next, select **Edit settings >**, next to the subscription for which you want to downgrade ASC.

4. On the next screen select "Pricing Tier" from the left-hand menu.

5. A new page will appear that looks like the image below. Click on the box on the left that says "Free (for Azure resources only)".



6. Press the **Save** button at the top of the screen.

You have now downgraded your subscription to the free tier of Azure Security Center. Congratulations, you have taken your first (and most important) step to securing your application, data and network!

Check your knowledge

1. True or false: Azure Security Center works for Azure resources and on-premises resources.

True

False

Check your answers

Inputs and Outputs

- 10 minutes

The most prevalent security weakness of current applications is a failure to correctly process data that is received from external sources, particularly *user input*. You should always take a close look at any input to make sure it has been validated before it is used. Failing to analyze user input for possible attacks can result in data loss or exposure, elevation of privilege, or even execution of malicious code on other users' computers.

The tragedy in this situation is that this scenario an easy problem to solve. In this unit we will cover how to treat data; when it's received, when it's displayed on the screen, and when it's stored for later use.

Why do we need to validate our input?

Imagine that you are building an interface to allow a user to create an account on your website. Our profile data includes a name, email, and a nickname that we will display to everyone who visits the site. What if a new user creates a profile and enters a nickname that includes some SQL commands? For example - what if a malicious user enters something like the following excerpt:
SQL

Copy

```
Eve'); DROP TABLE Users;--
```

If we blindly insert this value into a database, it could potentially alter the SQL statement to execute commands we absolutely don't want to run! This example is referred to as a "SQL Injection" attack, which is one of the *many* types of exploits that can potentially be done when you don't properly handle user input. So, what can we do to fix this situation? This unit will teach you when to validate input, how to encode output, and how to create parameterized queries (which solves the above exploit). These techniques are the three main defense techniques against malicious input being entered into your applications.

When do I need to validate input?

The answer is *always*. You must validate **every** input for your application. This includes parameters in the URL, input from the user, data from the database, data from an API and anything that is passed in the clear that a user could potentially manipulate. Always use an *allow list* approach, which means you only accept "known good" input, instead of a *deny list* (where you specifically look for bad input) because it's impossible to think of a complete list of potentially dangerous input. Do this work on the server, not the client-side (or in addition to the client-side), to ensure that your defenses cannot be circumvented. Treat **ALL** data as untrusted and you will protect yourself from most of the common web app vulnerabilities.
If you are using ASP.NET, the framework provides great support for validating input on both the client and server side.
If you are using another web framework, there are some great techniques for doing input validation available on the OWASP Input Validation Cheatsheet.

Always use parameterized queries

SQL databases are commonly used to store data; for example - your application could store user profile information in a database. Yous should never create inline SQL or other database queries in your code using raw user input and send it directly to the database; this behavior is a recipe for disaster, as we saw above.
For example - **do not** create code like the following inline SQL example:

C#

Copy

```
string userName = Request.QueryString["username"]; // receive input from
the user BEWARE!
...
string query = "SELECT *  FROM  [dbo].[users] WHERE userName = '" +
userName + "'";
```

Here we concatenate text strings together to create the query, taking the input from the user and generating a dynamic SQL query to look up the user. Again, if a malicious user realized we were doing this, or just *tried* different input styles to see if there was a vulnerability, we could end up with a major disaster. Instead, use parameterized SQL statements or stored procedures such as this:
SQL

Copy

```
-- Lookup a user
CREATE PROCEDURE sp_findUser
(
@UserName varchar(50)
)

SELECT *  FROM  [dbo].[users] WHERE userName = @UserName
```

With this method you can invoke the procedure from your code safely, passing it the `userName` string without worrying about it being treated as part of the SQL statement.

Always encode your output

Any output you present either visually or within a document should always be encoded and escaped. This can protect you in case something was missed in the sanitization pass, or the code accidentally generates something that can be used maliciously. This design principle will make sure that everything is displayed as *output* and not inadvertently interpreted as something that should be executed, which is another common attack technique that is referred to as "Cross-Site Scripting" (XSS).
Since XSS prevention is a common application requirement, this security technique is another area where ASP.NET will do the work for you. By default, all output is already encoded. If you are using another web framework, you can verify your options for output encoding on websites with the OWASP XSS Prevention Cheatsheet.

Summary

Sanitizing and validating your input is a necessary requirement to ensure your input is valid and safe to use and store. Most modern web frameworks offer built-in features that can automate some of this work. You can check your preferred framework's documentation and see what features it offers. While web applications are the most common place where this happens, keep in mind that other types of applications can be just as vulnerable. Don't think you're safe just because your new application is a desktop app. You will still need to properly handle user input to ensure someone doesn't use your app to corrupt your data, or damage your company's reputation.

Check your knowledge

1. Which of the following data sources need to be validated?

Data from a 3rd party API

Data from the URL parameter

Data collected from the user via an input field

All of the above

2. Parameterized queries (stored procedures in SQL) are a secure way to talk to the database because:

They are more organized than inline database commands, and therefore less confusing for users.

There is a clear outline of the script in the stored procedure, ensuring better visibility.

Parameterized queries substitute variables before running queries, meaning it avoids the opportunity for code to be submitted in place of a variable.

3. Which of the following data needs to be output encoded?

Data saved to the database

Data to be output to the screen

Data sent to a 3rd party API

Data in the URL parameters

Check your answers

输入和输出

- 10 分钟

当前应用程序最普遍的安全弱点在于无法正确处理从外部源接收的数据，特别是用户输入。 在使用任何输入之前，始终应该认真检查，确保它已经过验证。 如果未能分析用户输入中是否有潜在攻击，则可能导致数据丢失或泄露、提升特权，甚至在其他用户的计算机上执行恶意代码。

该情况中的悲剧在于这本是很容易解决的一个问题。 在本单元中，我们将介绍在数据接收时、数据在屏幕上显示时，以及在数据存储供稍后使用时如何处理数据。

为何需要验证输入?

假设你正在生成一个界面，使用户能够在你的网站上创建帐户。 个人资料数据包括姓名、电子邮件，以及要向该网站的任何访客显示的昵称。 如果某个新用户在创建个人资料时输入了包含一些 SQL 命令的昵称该怎么办? 例如，如果恶意用户输入下述引用之类的内容该怎么办:

SQL

复制

```
Eve'); DROP TABLE Users;--
```

如果我们草率地将此值插入数据库，它可能会修改 SQL 语句，使其执行我们绝对不想运行的命令！ 该示例被称作“SQL 注入”攻击，这是众多恶意利用类型中的一种，可能在你不当处理用户输入时发出。 那么，我们可如何弥补这一情况? 本单元将讲解何时要验证

输入、如何将输出编码，以及如何创建参数化的查询（这可以解决上述恶意利用问题！）。 这些方法就是防止在应用程序中输入恶意内容的三大主要防范方法。

何时需要验证输入?

答案是“始终”。 必须验证应用程序的每项输入。 这包括 URL 中的参数、用户的输入、数据库中的数据、API 中的数据，以及以明文形式传入的、用户可能要处理的任何内容。 请始终使用“允许列表”方法，它是指你只能接受“已知良好的”输入，而不是采用“拒绝列表”（此情况下会专门查找恶意输入），原因是没法想象出包含潜在恶意输入的完整列表。 在服务器而不是客户端上完成此工作（也可同时在客户端上完成此工作），确保不会绕过防御机制。 将所有数据视为不受信任，这样，在遇到大多数常见的 Web 应用漏洞时可以保护自己。

如果使用 ASP.NET，该框架可为客户端和服务器端的输入验证提供很好的支持。

如果使用其他 Web 框架，可以通过 OWASP 输入验证速查表中提供的一些极佳方法执行输入验证。

始终使用参数化查询

通常使用 SQL 数据库来存储数据；例如，你的应用程序可将用户个人资料信息存储在数据库中。 切勿在代码中使用原始用户输入来创建内联 SQL 或其他数据库，也不要将其直接发送到数据库中；此行为后患无穷，前面已对此进行过描述。

例如，不要创建类似于以下内联 SQL 示例的代码:

C#

复制

```
string userName = Request.QueryString["username"]; // receive input from the user BEWARE!
...
string query = "SELECT *  FROM  [dbo].[users] WHERE userName = '" + userName + "'";
```

在这里，我们将文本字符串连接起来创建查询，从用户处获取输入并生成一个动态 SQL 查询来查找用户。 同样，如果恶意用户意识到我们编写了这种代码，或者尝试不同的输入样式来确定是否存在漏洞，则我们最终可能会遭受重大灾难。 应使用参数化 SQL 语句或存储过程，如下所示:

SQL

复制

```
-- Lookup a user
CREATE PROCEDURE sp_findUser
(
@UserName varchar(50)
)

SELECT *  FROM  [dbo].[users] WHERE userName = @UserName
```

通过此方法，可以从代码安全调用过程，并向其传递 userName 字符串，而无需担心该过程被视为 SQL 语句的一部分。

始终将输出编码

以可视方式或者在文档中提供的任何输出应始终经过编码并转义。 如果在清理过程中遗失了数据，或者代码意外生成了可被恶意使用的内容，上述做法可以提供保护。 此设计原则将确保所有内容都显示为输出，且不会无意中被解释为应执行的内容，后者就是另一种被称作“跨站点脚本”(XSS) 的常见攻击技术。
由于 XSS 防范是一种常见的应用程序要求，因此该安全方法也由 ASP.NET 代为操作。 默认情况下，所有输出均已编码。 如果使用其他 Web 框架，可以使用 OWASP XSS 防护速查表来验证网站上用于输出编码的选项。

总结

必须清理和验证你的输入以确保该输入有效且可安全使用和存储。 大多数现代 Web 框架都提供了可自动处理其中部分操作的内置功能。 可查看首选框架的文档，了解它所提供的功能。 虽然最常在 Web 应用中进行此操作，但请记住，其他类型的应用程序也可能同样易受攻击。 不要因为新应用程序是桌面应用程序就认为自己是安全的。 仍需正确处理用户输入，以确保有人不会利用你的应用来破坏数据，或损害公司的声誉。

知识检查

1. 需要验证下面哪些数据源?

第三方 API 中的数据

URL 参数中的数据

通过输入字段从用户收集的数据

以上都是
2. 参数化查询（SQL 中的存储过程）是与数据库通信的安全方式，因为：

它们比内联数据库命令更有条理，因此，较不容易给用户造成混淆。

存储过程中的脚本具有明确的大纲，可确保更好的直观性。

参数化查询在运行查询之前会替换变量，这意味着，可以避免取代变量提交代码的可能性。
3. 需要对以下哪些数据进行输出编码?

保存到数据库的数据

要输出到屏幕的数据

发送到第三方 API 的数据

URL 参数中的数据
检查你的答案

密钥保管库中的

机
密

- 5 分钟

如果将机密与所有人共享，机密就不是机密了。 将连接字符串、安全令牌、证书和密码等机密项目存储在代码中相当于邀请某人去取用它们，而取用的目的并不是你的本来目的。 将此类数据存储在 Web 配置中也不妥 - 这实际上是允许能够访问源代码或 Web 服务器的人访问你的专用数据。

与上述做法相反，应始终将这些机密置于 **Azure Key Vault** 中。

什么是 Azure Key Vault

Azure Key Vault 是一个机密存储：用于存储应用程序机密的集中式云服务。 Key Vault 将应用程序机密保存在一个中心位置，并提供安全访问、权限控制和访问日志记录，以这种方式来确保机密数据的安全。

机密存储在各个保管库中，每个保管库都有自己的配置和安全策略，用于控制访问。 然后，你可以通过 REST API 或者通过为大多数语言提供的客户端 SDK 访问数据。

重要

Key Vault 旨在存储服务器应用程序的配置机密。 它并非用于存储属于应用用户的数据，也不应该用于应用的客户端部分。 这反映在其性能特征、API 和成本模型中。

用户数据应存储在其他位置，例如使用透明数据加密的 Azure SQL 数据库或使用存储服务加密的存储帐户。 应用程序用于访问这些数据存储的机密可保存在 Key Vault 中。

为何使用 Key Vault 来存储机密

密钥管理和机密存储操作在手动进行的情况下可能很复杂，也容易出错。 手动轮换证书意味着可能需要经历数小时或数天没有证书的情况。 如上所述，将连接字符串保存在配置文件或代码存储库中意味着某人可能会窃取你的凭证。
Key Vault 允许用户存储连接字符串、机密、密码、证书、访问策略、文件锁（使 Azure 中的项目只读）和自动化脚本。 它还记录访问和活动，允许你监视订阅中的访问控制（标识和访问管理），并且有诊断、指标、警报和故障排除工具，可以确保你有所需的访问权限。
有关使用 Azure Key Vault 的详细信息，请参阅使用 Azure Key Vault 管理服务器应用中的机密。

总结

如果使用 Azure Key Vault 正确管理机密，则根本不用担心凭证被窃、手动密钥轮换和证书续订等事项。

下一单元: 框架更新

S
e
c
r

etsinKeyVault

- 5 minutes

Secrets aren't secrets if they are shared with everyone. Storing confidential items like connection strings, security tokens, certificates and passwords in your code is just inviting someone to take them and use them for something other than what you intended them for. Even storing this sort of data in your web configuration is a bad idea - you are essentially allowing anyone who has access to the source code or web server access to your private data.

Instead, you should always put these secrets into **Azure Key Vault**.

What is Azure Key Vault

Azure Key Vault is a *secret store*: a centralized cloud service for storing application secrets. Key Vault keeps your confidential data safe by keeping application secrets in a single central location and providing secure access, permissions control, and access logging.
Secrets are stored in individual *vaults*, each with their own configuration and security policies to control access. You can then get to your data through a REST API, or through a client SDK available for most languages.

Important

Key Vault is designed to store configuration secrets for server applications. It's not intended for storing data belonging to your app's users, and it shouldn't be used in the client-side part of an app. This is reflected in its performance characteristics, API, and cost model.
User data should be stored elsewhere, such as in an Azure SQL database with Transparent Data Encryption, or a storage account with Storage Service Encryption. Secrets used by your application to access those data stores can be kept in Key Vault.

Why use a Key Vault for my secrets

Key management and storing secrets can be complicated and error-prone when performed manually. Rotating certificates manually means potentially going without for a few hours, or days. As mentioned above, saving your connections strings in your configuration file or code repository means someone could steal your credentials.
Key Vault allows users to store connection strings, secrets, passwords, certificates, access policies, file locks (making items in Azure read-only), and automation scripts. It also logs access and activity, allows you to monitor access control (IAM) in your subscription, and it also has diagnostic, metrics, alerts and troubleshooting tools, to ensure you have the access you need.
Learn more about using an Azure Key Vault in Manage secrets in your server apps with Azure Key Vault.

Summary

Credential theft, manual key rotation and certificate renewal can be a thing of the past if you manage your secrets well, using Azure Key Vault.

F
r
a
m
e
w
o
r
k
U
p
d
a
t

e

s

- 7 minutes

Many developers consider the frameworks and libraries they use to build their software with to be primarily decided by features or personal preference. However, the framework that you choose is an important decision, not only from a design and functionality perspective but also from a *security* perspective. Choosing a framework with modern security features and keeping it up-to-date is one of the best ways to ensure your apps are secure.

Choose your framework carefully

The most important factor regarding security when choosing a framework is how well supported it is. The best frameworks have stated security arrangements and are supported by large communities who improve and test the framework. No software is 100% bug-free or totally secure, but when a vulnerability is identified, we want to be certain that it will be closed or have a workaround provided quickly.

Often "well supported" is synonymous with "modern". Older frameworks tend to either be replaced or eventually fade in popularity. Even if you have significant experience with (or many apps written in) an older framework, you'll be better off choosing a modern library that has the features you need. Modern frameworks tend to build on the lessons learned by earlier iterations which makes choosing them for new apps a form of threat surface reduction. You will have one more app to worry about if a vulnerability is discovered in the older framework that your legacy applications are written in.

For more information on secure design and reducing threat surface, see Design For Security in Azure.

Keep your framework updated

Software development frameworks, such as Java Spring and .NET Core release updates and new versions regularly. These updates include new features, removal of old features, and often security fixes or improvements. When we allow our frameworks to become out of date, it creates "technical debt". The further out of date we get, the harder and riskier it will be to bring our code up to the latest version. In addition, much like the initial framework choice, staying on older versions of the framework open you up to more security threats which have been fixed in newer releases of the framework.
As an example, from 2016-2017, over 30 vulnerabilities were found in the Apache Struts framework. These were quickly addressed by the development team, but some companies didn't apply the patches and paid the price in the form of a data breach. **Make sure to keep your frameworks and libraries up-to-date**.

How do I update my framework?

Some frameworks, like Java or .NET, require an install and tend to release on a known cadence. It's a good idea to watch for new releases and plan to make a branch of your code to try it out when it's released. As an example, .NET Core maintains a release notes page which you can check to find the latest versions available.
More specialized libraries such as JavaScript frameworks, or .NET components can be updated through a package manager. **NPM** and **Webpack** are popular choices for web projects and are supported by most IDEs or build tools. In .NET, we use **NuGet** to manage our component dependencies. Much like updating the core framework, branching your code, updating the components and testing is a good technique to validate a new version of a dependency.

Note

The `dotnet` command-line tool has an `add package` and `remove package` option to add or remove NuGet packages but doesn't

have a corresponding `update package` command. However, it turns out you can run `dotnet add package <package-name>` in your project and it will automatically *upgrade* the package to the latest version. This is an easy way to update dependencies without having to open the IDE.

Take advantage of built-in security

Always check to see what security features your frameworks offer. **Never** roll your own security if there's a standard technique or capability built in. In addition, rely on proven algorithms and workflows because these have often been scrutinized by many experts, critiqued and strengthened so you can be assured that they are reliable and secure.
The .NET Core framework has countless security features, here are a few core starting places in the documentation.

- Authentication -Identity Management
- Authorization
- Data Protection
- Secure Configuration
- Security Extensibility APIs

Each one of these features was written by experts in their field, and then battered with tests to ensure that it works as intended, and only as intended. Other frameworks offer similar features - check with the vendor that provides the framework to find out what they have in each category.

Warning

Writing your own security controls, instead of using those provided by your framework, is not only wasting time, it's less secure.

Azure Security Center

When using Azure to host your web applications, Security Center will warn you if your frameworks are out of date as part of the recommendations tab. Don't forget to look there from time to time to see if there are any warnings related to your apps.

Use the latest supported PHP version for Web Application (Preview)	20 web applications
Use the latest supported Node.js version for Web Application (Preview)	3 web applications

Summary

Whenever possible, choose a modern framework to build your apps, always use the built-in security features, and make sure you keep it up-to-date. These simple rules will help to ensure your application starts on a solid foundation.

Next unit: Safe Dependencies

Continue

框架更新

- 7 分钟

许多开发人员主要是根据功能和个人偏好来决定使用哪种框架和库来构建软件。 但是，选择框架是一个重要决策，不仅要考虑到设计和功能，而且要考虑到安全方面。 选择具有新式安全功能的框架并使其保持最新，是确保应用安全的最佳方式之一。

谨慎选择框架

选择框架时，最重要的安全因素是该框架的受支持力度如何。 最佳的框架有明确的安全规划，并受到致力于改进和测试框架的大社区的支持。 任何软件都不可能做到完全无 bug 或绝对安全，但识别漏洞后，我们需要确保快速解决该漏洞或提供解决方法。
通常，“新式”就意味着“良好的支持”。 旧式框架会逐步被取代，或者最终丧失普及性。 即使你对使用旧式框架（编写的许多应用）拥有丰富的经验，也最好是选择具有所需功能的新式库。 新式框架往往构建在过去更新换代时学到的经验教训基础之上，为新应用选择这种框架可以减少威胁面。 在用于编写旧版应用程序的旧式框架中发现漏洞时，会多一个需要担心的应用。
有关安全设计和减少威胁面的详细信息，请参阅 Azure 中的安全设计。

保持框架处于最新状态

Java Spring 和 .NET Core 等软件开发框架会定期发布更新和新版本。 这些更新会包含新功能并删除旧功能，并且通常会包含安全修补程序或改进。 如果允许框架过时，则会产生“技术债务”。 框架越过时，发布代码最新版本的难度就越高，且风险越大。 此外，与初始选择框架时一样，沿用旧版框架会曝露在更多的安全威胁之中，而新版框架会解决这些威胁。
例如，在 2016 年到 2017 年，我们在 Apache Struts 框架中发现了 30 多个漏洞。 开发团队很快就解决这些漏洞，但有些公司未应用修补程序，数据违规让它们付出了代价。 **请务必保持框架和库的最新状态**。

如何更新框架?

某些框架（例如 Java 或 .NET）需要安装，并按已知的频率发布。 最好是先等待新版本，并计划好创建代码的分支，以便在新

版本发布时试用它。 例如，.NET Core 维护一个发行说明页，可在其中检查是否有最新版本可用。
JavaScript 框架或 .NET 组件等更专业的库可通过包管理器进行更新。 NPM 和 Webpack 是适合 Web 项目的热门选项，受大多数 IDE 或生成工具支持。 在 .NET 中，我们使用 NuGet 来管理组件依赖项。 与更新核心框架一样，创建代码的分支、更新组件并进行测试是验证依赖项新版本的不错手段。

备注

`dotnet` 命令行工具提供 `add package` 和 `remove package` 选项用于添加或删除 NuGet 包，但没有对应的 `update package` 命令。 但是，可以改为在项目中运行 `dotnet add package <package-name>`，它会自动将包升级到最新版本。 这是更新依赖项的简便方法，无需打开 IDE。

利用内置安全性

始终检查框架提供哪些安全功能。 如果有标准的技术或内置功能，**切勿**实施自己的安全功能。 此外，请依赖成熟的算法和工作流，因为它们通常已经过专家的审查、评判和增强，可以确信它们是可靠且安全的。
.NET Core 框架提供无数的安全功能，下面是在文档中介绍过的几个核心入门功能。

- 身份验证 - 标识管理
- 授权
- 数据保护
- 安全配置
- 安全扩展性 API

其中的每项功能由领域专家编写，并通过全方位的测试，确保它按预期方式工作，并且只会按意图工作。 其他框架提供类似功能 - 请咨询框架的供应商，了解他们在每个类别中所提供的功能。

警告

如果自行编写安全控件，而不使用框架提供的代码，这不仅浪费时间，而且不太安全。

Azure 安全中心

使用 Azure 托管 Web 应用程序时，如果框架过时，则安全中心会在“建议”选项卡中发出警告。请记得不时查看是否出现了与你的应用相关的任何警告。

总结

尽量选择新式框架来构建应用，始终使用内置安全功能，并确保保持框架的最新状态。 这些简单的规则将帮助确保你的应用程序具有坚实的基础。

下一单元: 安全依赖项

继续

安全依

赖
项

- 5 分钟

现代应用程序中很大一部分代码是由开发人员选择的库和依赖关系组成的。 这是一种常见的做法，既省时间又省金钱。 但是，缺点是你现在要为该代码负责，因为你在项目中使用了它，即使它是别人编写的。 如果某位研究人员（或黑客，这会更糟）在某个这样的第三方库中发现一个漏洞，则同样的缺陷也可能会存在于你的应用中。

在我们的行业中，使用包含已知漏洞的组件是一个大问题。 此问题很严重，以至于多年来一直高居最严重 Web 应用程序漏洞 OWASP 前 10 的位置，占据第 9 位。

跟踪已知安全漏洞

我们的问题是了解何时发现问题。 始终更新库和依赖项（在我们的列表中排第 4）当然会有用，但最好是跟踪已确定的可能会影响应用程序的漏洞。

重要

当系统存在已知漏洞时，很可能漏洞利用工具（可以用来攻击这些系统的代码）也已经存在了。 如果漏洞利用工具已公开，则必须立即更新任何受影响的系统。

Mitre 是一家非盈利性的组织，负责维护常见漏洞和风险列表。 该列表是一组可以公开搜索的已知网络安全漏洞，这些漏洞存在于应用、库和框架中。 **如果在 CVE 数据库中发现一个库或组件，则该库或组件有已知漏洞**。

在产品或组件中发现安全漏洞后，安全社区会提交问题。 每个发布的问题都会分配一个 ID，并且会包含发现日期、漏洞说明、对已发布解决方法或有关问题的供应商声明的引用。

如何验证第三方组件中是否有已知漏洞

你可以将日常任务置于手机中，以便检查此列表。不过对我们来说，幸运的是存在许多可以用来验证依赖项是否有漏洞的工具。可以对代码库运行这些工具，或者在开发过程中将其添加到 CI/CD 管道，以便自动检测是否存在问题，这样会更好。

- OWASP 依赖项检查，其中有 Jenkins 插件
- OWASP SonarQube
- Synk：对 GitHub 中的开源存储库免费提供
- Black Duck：得到众多企业的运用
- RubySec：一个仅适用于 Ruby 的咨询数据库
- Retire.js：一项用于验证 JavaScript 库是否过时的工具；可以用作各种工具（包括 Burp Suite）的插件

某些专用于静态代码分析的工具也可用于此目的。

- Roslyn Security Guard
- Puma Scan
- PT Application Inspector
- Apache Maven 依赖项插件
- Source Clear
- Sonatype
- Node Security Platform
- WhiteSource
- Hdiv
- 还有更多...

若要详细了解使用含漏洞插件的风险，请访问专门介绍此主题的 OWASP 页。

摘要

在应用程序中使用库或其他第三方组件时，也要承受其所带来的风险。 若要降低此风险，最好的方法是确保只使用没有已知漏洞的组件。

下一单元: 结束语

继续

SafeDependenc

i

e

s

- 5 minutes

A large percentage of code present in modern applications is made up of the libraries and dependencies chosen by you: the developer. This is a common practice that saves time and money. However, the downside is that you are now responsible for this code, even though others wrote it, because you used it in your project. If a researcher (or worse, a hacker) discovers a vulnerability in one of these third-party libraries, then the same flaw will likely also be present in your app.

Using components with known vulnerabilities is a huge problem in our industry. It is so problematic that it has made the OWASP top 10 list of worst web application vulnerabilities, holding at #9 for several years.

Track known security vulnerabilities

The problem we have is knowing when an issue is discovered. Keeping our libraries and dependencies updated (#4 in our list!) will of course help, but it's a good idea to keep track of identified vulnerabilities that might impact your application.

Important

When a system has a known vulnerability, it is much more likely also to have exploits available, code that people can use to attack those systems. If an exploit is made public, it is crucial that any affected systems are updated immediately.

Mitre is a non-profit organization that maintains the Common Vulnerabilities and Exposures list. This list is a publicly searchable set of known cybersecurity vulnerabilities in apps, libraries, and frameworks. **If you find a library or component in the CVE database, it has known vulnerabilities**.

Issues are submitted by the security community when a security flaw is found in a product or component. Each published issue is assigned an ID and contains the date discovered, a description of the vulnerability, and references to published workarounds or vendor statements about the issue.

How to verify if you have known vulnerabilities in your third-party components

You could put a daily task into your phone to check this list, but luckily for us, many tools exist to allow us to verify if our dependencies are vulnerable. You can run these tools against your codebase, or better yet, add them to your CI/CD pipeline to automatically check for issues as part of the development process.

- OWASP Dependency Check, which has a Jenkins plugin
- OWASP SonarQube
- Snyk, which is free for open-source repositories in GitHub
- Black Duck which is used by many enterprises
- RubySec an advisory database just for Ruby
- Retire.js a tool for verifying if your JavaScript libraries are out of date; can be used as a plugin for various tools, including Burp Suite

Some tools made specifically for static code analysis can be used for this as well.

- Roslyn Security Guard
- Puma Scan
- PT Application Inspector
- Apache Maven Dependency Plugin
- Source Clear
- Sonatype
- Node Security Platform
- WhiteSource
- Hdiv

- And many more...

For more information on the risks involved in using vulnerable components visit the OWASP page dedicated to this topic.

Summary

When you use libraries or other third-party components as part of your application, you are also taking on any risks they may have. The best way to reduce this risk is to ensure that you are only using components that have no known vulnerabilities associated with them.