第 4 章 治理、风险和合规性

随着数字环境变得越来越错综复杂，管理网络安全风险和维护合规性也变得越来越具有挑战性，这被称为治理、风险和合规性（Governance, Risk, and Compliance，GRC）难题。本章将探讨如何结合使用 ChatGPT 和 OpenAI API 的强大功能，为此问题提供富有洞察力的解决方案，从而显著提高网络安全基础设施的效率和有效性。

本章将探索如何利用 ChatGPT 的功能来生成全面的网络安全策略，从而简化创建策略的复杂任务。我们将为你介绍一种创新方法，允许对策略文件的每一部分进行精细控制，提供一个强大的网络安全框架，以满足你的特定业务需求。

在此基础上，我们将深入研究复杂网络安全标准的细微差别。ChatGPT 对此起到了指导作用，它可以将复杂的合规性要求分解为可管理的、清晰的步骤，从而为确保标准合规性提供一条简明通畅的途径。

此外，我们还将探索网络风险评估的关键领域，揭示自动化如何革新这一重要过程。你将深入了解识别潜在威胁、评估漏洞和建议适当的控制措施，从而大大提高组织管理网络安全风险的能力。

在风险评估之后，我们的讨论重点将转向有效地对这些风险进行优先级排序。你将了解 ChatGPT 如何帮助创建基于各种风险相关因素的客观评分算法，这将使你能够战略性地分配资源来管理具有最高优先级的风险。

最后，我们将讨论生成风险报告的基本任务。详细的风险评估报告不仅可以作为已识别风险和缓解策略的宝贵记录，还可以确保利益相关者之间的清晰沟通。我们将展示如何使用 ChatGPT 自动创建此类报告，这将帮助你节省时间并保持所有文档的一致性。

本章包含以下秘笈：

● 安全策略和程序生成

● 网络安全标准合规性

● 创建风险评估流程

● 风险排序和优先级

● 构建风险评估报告

4.1 技 术 要 求

本章需要一个 Web 浏览器和稳定的互联网连接来访问 ChatGPT 平台并设置你的账户。你还需要设置你的 OpenAI 账户，并获得你的 API 密钥。如果没有，请参考第 1 章“基础知识介绍：ChatGPT、OpenAI API 和提示工程”以了解详细信息。

此外，你还需要对 Python 编程语言有一个基本的了解，并且会使用命令行，因为你将使用 Python 3.x，它需要安装在你的系统上，以便你可以使用 OpenAI GPT API 并创建 Python 脚本。

最后，你还需要一个代码编辑器，这对于编写和编辑本章中的 Python 代码和提示文件也是必不可少的。

本章的代码文件可在以下网址找到：

https://github.com/PacktPublishing/ChatGPT-for-Cybersecurity-Cookbook

4.2 安全策略和程序生成

本秘笈将利用 ChatGPT 和 OpenAI API 的功能为你的组织生成全面的网络安全策略。这一过程对于 IT 经理、首席信息安全官（chief information security officer，CISO）和网络安全专业人员来说是非常宝贵的，因为他们都希望创建一个强大的网络安全框架，以适应自己的特定业务需求。

在本秘笈中，你给 ChatGPT 分配的角色是一名经验丰富的网络安全专业人员，专门负责治理、风险和合规性（Governance, Risk, and Compliance，GRC）问题。你将学习如何使用 ChatGPT 生成结构良好的策略大纲，然后通过后续提示不断填充和丰富大纲的每个部分。这种方法使你能够生成全面完善的文档，并对每个部分进行细粒度控制，而不必考虑 ChatGPT 的 token 限制和上下文窗口问题。

此外，本秘笈还将介绍如何使用 OpenAI API 和 Python 自动执行策略生成过程，随后将网络安全策略生成为 Microsoft Word 文档。这一指南将提供一个实用的框架，用于使用 ChatGPT 和 OpenAI API 制定详细且量身定制的网络安全策略。

4.2.1 准备工作

在深入学习本秘笈之前，请确保你的 OpenAI 账户已设置，并且你可以访问你的 API 密钥。如果你还没有设置或需要复习，则请回到第 1 章“基础知识介绍：ChatGPT、OpenAI API 和提示工程”。

你还需要确认安装了以下 Python 库：

（1）openai：这是官方的 OpenAI API 客户端库，我们将使用它与 OpenAI API 进行交互。其安装命令如下：

**pip install openai**

（2）os：这是一个内置的 Python 库，所以不需要安装。我们将使用它与操作系统交互，特别是从你的环境变量中获取 OpenAI API 密钥。

（3）python-docx：这是一个用于创建 Microsoft Word 文档的 Python 库。其安装命令如下：

**pip install python-docx**

（4）markdown：该库可用于将 Markdown 转换为 HTML，对于生成格式化文档很有用。其安装命令如下：

**pip install markdown**

（5）tqdm：该库用于显示策略生成过程中的进度条。其安装命令如下：

**pip install tqdm**

一旦所有这些要求得到了满足，你就可以开始使用 ChatGPT 和 OpenAI API 生成网络安全策略了。

4.2.2 实战操作

本节将指导你使用 ChatGPT 生成符合你所在组织需求的详细网络安全策略。通过提供必要的详细信息并使用给定的系统角色和提示，将生成结构良好的网络安全策略文件。

请按以下步骤操作：

（1）登录你的 OpenAI 账户并导航到 ChatGPT Web 用户界面。

（2）单击 New chat（新建聊天）按钮，启动与 ChatGPT 的新对话。

（3）输入以下系统角色以设置 ChatGPT 的上下文：

You are a cybersecurity professional specializing in governance,

risk, and compliance (GRC) with more than 25 years of

experience.

（4）输入以下消息文本，请注意将大括号（｛｝）中的占位符替换为基于组织需要的相关信息。你可以将此提示与系统角色结合使用，也可以按如下方式单独输入（注意将公司名称和类型替换为你自己公司的名称和类型）：

Write a detailed cybersecurity policy outline for my company,

{company name}, which is credit union. Provide the outline only,

with no context or narrative. Use markdown language to denote

the proper headings, lists, formatting, etc.

（5）查看 ChatGPT 的输出。如果它令人满意并且符合你的要求，则可以继续下一步。如果不满意，则可以选择细化提示或再次运行对话以生成不同的输出。

（6）根据大纲生成策略。对于大纲的每个部分，可使用以下内容提示 ChatGPT，请注意将｛section｝替换为大纲中相应的部分（节）标题：

You are currently writing a cybersecurity policy. Write the

narrative, context, and details for the following section

(and only this section): {section}. Use as much detail and

explanation as possible. Do not write anything that should go in

another section of the policy.

（7）一旦获得了所需的输出，可将生成的响应直接复制并粘贴到 Word 文档或你选择的编辑器中，以创建一个全面的网络安全策略文档。

4.2.3 原理解释

这个 GPT 辅助的网络安全策略创建秘笈利用了自然语言处理和机器学习算法的力量，它可以为你量身制定一个全面网络安全策略，以满足你组织的需求。

通过给 ChatGPT 分配特定的系统角色，并利用详细的用户请求作为提示，网络安全专业人员将能够调整其输出，以满足自己的需求，生成详细的策略。

该过程的工作原理如下：

（1）系统角色和详细提示：本示例为 ChatGPT 分配的系统角色是一名经验丰富的网络安全专业人士，专门负责 GRC 问题。输入的提示作为用户请求，详细描述了策略大纲的细节，内容涉及公司的性质和网络安全策略的要求等。这些输入提供了上下文并引导 ChatGPT 的响应，确保其满足策略创建任务的复杂性和要求。

（2）自然语言处理（NLP）和机器学习（ML）：NLP 和 ML 是 ChatGPT 能力的基础。它使用这些技术来理解用户请求的复杂性，从模式中学习，并生成一个结构良好的详细、具体且全面的网络安全策略。

（3）知识和语言理解能力：ChatGPT 可以利用其庞大的知识库和语言理解功能，遵守行业标准方法和最佳实践。这在迅速发展的网络安全领域至关重要，它将确保生成的网络安全策略是最新的，并符合公认的标准。

（4）迭代策略生成：在根据生成的大纲创建详细策略的过程中，需要对策略的每个部分迭代提示 ChatGPT。这允许你对每个部分的内容进行更精细的控制，有助于确保策略的结构和组织良好。

（5）简化策略创建过程：利用 GPT 辅助方式创建网络安全策略的总体好处是，它简化了创建全面网络安全策略的过程，减少了用户在创建策略上花费的时间，并允许生成符合行业标准和组织特定需求的专业级别策略。

通过使用这些详细的输入，你可以将 ChatGPT 转变为一个潜在的宝贵工具，帮助你创建一个详尽的、量身定制的网络安全策略。这不仅可以增强你的网络安全态势，还可以确保资源被有效地用于保护你的组织。

4.2.4 扩展知识

在上述 ChatGPT 秘笈的基础上，你还可以使用 OpenAI API 来增强其功能，使得它不仅可以生成网络安全策略大纲，还可以填充和丰富每个部分的详细信息。当你希望即时创建详细的文档或为具有不同需求的多家公司生成策略时，这种方法非常有用。

本小节中的 Python 脚本包含了与 ChatGPT 版本相同的思路，但通过 OpenAI API 提供了附加功能，可以对内容生成过程进行更多的控制。

要使用 OpenAI API 生成网络安全策略，请按以下步骤操作：

（1）导入必要的库并设置 OpenAI API：

import os

import openai

from openai import OpenAI

import docx

from markdown import markdown

from tqdm import tqdm

# get the OpenAI API key from environment variable

openai.api\_key = os.getenv('OPENAI\_API\_KEY')

在这一步中，我们导入了所需的库，如 openai、os、docx、markdown 和 tqdm。此外，还通过提供 API 密钥设置了 OpenAI API。

（2）为网络安全策略大纲准备初始提示：

# prepare initial prompt

messages=[

{

"role": "system",

"content": "You are a cybersecurity

professional specializing in governance,

risk, and compliance (GRC) with more than

25 years of experience."

},

{

"role": "user",

"content": "Write a detailed cybersecurity

policy outline for my company,

{company name}, which is a credit union.

Provide the outline only, with no context

or narrative. Use markdown language to

denote the proper headings, lists,

formatting, etc."

}

]

可以看到，该初始提示是使用两个角色的对话构建的，这两个角色就是 system 和 user。system 消息设置了上下文，告诉人工智能模型，为它分配的角色是一个经验丰富的网络安全专业人员。user 消息则指示人工智能模型需要为一家信用合作社（credit union）创建网络安全策略大纲，并指定需要 Markdown 格式的响应。

（3）使用 OpenAI API 生成网络安全策略大纲：

print("Generating policy outline...")

try:

client = OpenAI()

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

except Exception as e:

print("An error occurred while connecting to the

OpenAI API:", e)

exit(1)

# get outline

outline =

response.choices[0].message.content.strip()

print(outline + "\n")

上述代码可以将请求发送到 OpenAI API，并在完成后检索已生成的策略大纲。

（4）将大纲分成几个部分，并准备一份 Word 文档：

# split outline into sections

sections = outline.split("\n\n")

# prepare Word document

doc = docx.Document()

html\_text = ""

上述代码可将大纲分为若干个不同的部分（节），每个部分包含一个 Markdown 格式的标题或副标题。然后，使用 docx.Document()函数初始化一个新的 Word 文档。

（5）在大纲的每个部分上循环迭代，以生成详细信息：

# for each section in the outline

for i, section in tqdm(enumerate(sections, start=1),

total=len(sections), leave=False):

print(f"\nGenerating details for section {i}...")

上述代码将循环浏览大纲的每个部分。tqdm 函数用于显示进度条。

（6）为 AI 模型准备提示，以生成当前部分的详细信息：

# prepare prompt for detailed info

messages=[

{

"role": "system",

"content": "You are a cybersecurity

professional specializing in

governance, risk, and compliance (GRC)

with more than 25 years of

experience."

},

{

"role": "user",

"content": f"You are currently writing a

cybersecurity policy. Write the

narrative, context, and details for

the following section (and only this

section): {section}. Use as much

detail and explanation as possible. Do

not write anything that should go in

another section of the policy."

}

]

（7）生成当前部分的详细信息并将其添加到 Word 文档中：

try:

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

except Exception as e:

print("An error occurred while connecting to

the OpenAI API:", e)

exit(1)

# get detailed info

detailed\_info =

response.choices[0].message.content.strip()

# convert markdown to Word formatting

doc.add\_paragraph(detailed\_info)

doc.add\_paragraph("\n") # add extra line break

for readability

# convert markdown to HTML and add to the html\_text string

html\_text += markdown(detailed\_info)

上述代码使用了 OpenAI API 为当前部分生成详细信息。Markdown 格式的文本将转换为 Word 格式并添加到 Word 文档。此外，它还将被转换为 HTML 并添加到 HTML\_text 字符串。

（8）保存 Word 和 HTML 文档的当前状态：

# save Word document

print("Saving sections...")

doc.save("Cybersecurity\_Policy.docx")

# save HTML document

with open("Cybersecurity\_Policy.html", 'w') as f:

f.write(html\_text)

Word 文档和 HTML 文档的当前状态将在处理完每个部分后保存。这样可以确保在脚本中断时不会丢失任何进度。

（9）在处理完所有部分后打印完成消息：

print("\nDone.")

完整脚本如下：

import os

import openai

from openai import OpenAI

import docx

from markdown import markdown

from tqdm import tqdm

# get the OpenAI API key from environment variable

openai.api\_key = os.getenv('OPENAI\_API\_KEY')

# prepare initial prompt

messages=[

{

"role": "system",

"content": "You are a cybersecurity professional

specializing in governance, risk, and

compliance (GRC) with more than 25 years of

experience."

},

{

"role": "user",

"content": "Write a detailed cybersecurity policy

outline for my company, XYZ Corp., which is a

credit union. Provide the outline only, with no

context or narrative. Use markdown language to

denote the proper headings, lists, formatting,

etc."

}

]

print("Generating policy outline...")

try:

client = OpenAI()

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

except Exception as e:

print("An error occurred while connecting to the OpenAI

API:", e)

exit(1)

# get outline

outline =

response.choices[0].message.content.strip()

print(outline + "\n")

# split outline into sections

sections = outline.split("\n\n")

# prepare Word document

doc = docx.Document()

html\_text = ""

# for each section in the outline

for i, section in tqdm(enumerate(sections, start=1),

total=len(sections), leave=False):

print(f"\nGenerating details for section {i}...")

# prepare prompt for detailed info

messages=[

{

"role": "system",

"content": "You are a cybersecurity

professional specializing in governance,

risk, and compliance (GRC) with more than

25 years of experience."

},

{

"role": "user",

"content": f"You are currently writing a

cybersecurity policy. Write the narrative,

context, and details for the following

section (and only this section): {section}.

Use as much detail and explanation as

possible. Do not write anything that should

go in another section of the policy."

}

]

try:

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

except Exception as e:

print("An error occurred while connecting to the

OpenAI API:", e)

exit(1)

# get detailed info

detailed\_info =

response.choices[0].message.content.strip()

# convert markdown to Word formatting

doc.add\_paragraph(detailed\_info)

doc.add\_paragraph("\n") # add extra line break for

readability

# convert markdown to HTML and add to the html\_text

string

html\_text += markdown(detailed\_info)

# save Word document

print("Saving sections...")

doc.save("Cybersecurity\_Policy.docx")

# save HTML document

with open("Cybersecurity\_Policy.html", 'w') as f:

f.write(html\_text)

print("\nDone.")

这个 Python 脚本自动化了为特定公司 XYZ Corp.（一家信用合作社）生成详细网络安全策略大纲的过程。该脚本导入了必要的库，设置了 OpenAI API 密钥，并为 AI 模型准备了初始提示，指示其生成策略大纲。

在收到来自 OpenAI API 的成功响应后，脚本打印出策略大纲，并将其分解为单独的部分以进一步详细说明。脚本启动了 Word 文档来记录这些详细信息，然后，脚本在策略大纲的每个部分上迭代循环，生成详细信息并将其从 OpenAI API 附加到 Word 文档和 HTML 字符串，从而有效地创建了 Word 和 HTML 格式的详细策略文档。

每次迭代后，脚本都会确保文档得到保存，从而提供了一个安全网，防止因中断而导致数据丢失。在所有部分的循环迭代都保存文档之后，意味着策略大纲的详细信息创建完成。因此，在使用 OpenAI API 和 Python 自动化的过程中，粗略的策略大纲即被扩展为详细、全面的网络安全策略。

4.3 网络安全标准合规性

本秘笈将指导你如何使用 ChatGPT 来协助实现网络安全标准的合规性。

由于网络安全标准的编写方式，理解这些标准的要求可能很复杂。有了 ChatGPT 的帮助，即可简化此任务。你可以将网络安全标准的要求作为提示提供给 ChatGPT，然后模型可以将这些要求分解为更简单的术语，并帮助确认你是否合规，如果不合规，那么需要采取哪些步骤才能满足合规性要求。

4.3.1 准备工作

登录你的 OpenAI 账户，确保你可以访问 ChatGPT 界面。准备一份网络安全标准文件，以供你引用其中的要求。

4.3.2 实战操作

为了利用 ChatGPT 理解和检查是否符合网络安全标准，请执行以下步骤：

（1）登录 ChatGPT 界面。

（2）使用以下提示为 ChatGPT 分配角色：

You are a cybersecurity professional and CISO with 30 years of

experience in the industrial cybersecurity industry.

请注意，你应该用你所在的行业来取代上述提示中的 industrial。

（3）向 ChatGPT 提供你的提示：

"I need your help understanding the requirements of the NIST

SP 800-82 revision 2 standard. I will give you the requirement

ID, specifications, and any other supplemental information

I have that is associated with the requirement. You will

then explain the requirement to me in way that is easier to

understand, and form a question based on the requirement to

help me determine whether or not I comply with that requirement

or not. You will follow up by asking me if I have any further

questions about that requirement or if I'm ready to move to

the next requirement. If I have another question regarding

that requirement, we will continue discussing the requirement

in this manner. If I tell you I'm ready to move on to the next

requirement, we will start this process again with the next

requirement."

请记住将NIST SP 800-82 revision 2 standard标准名称替换为你使用的网络安全标准。

（4）向 ChatGPT 提供第一个需求的 ID、规范和任何补充信息，如图 4.1 所示。

（5）根据 ChatGPT 提供的响应与它进行对话，要么深入研究特定需求，要么继续下一个需求，如图 4.2 所示。

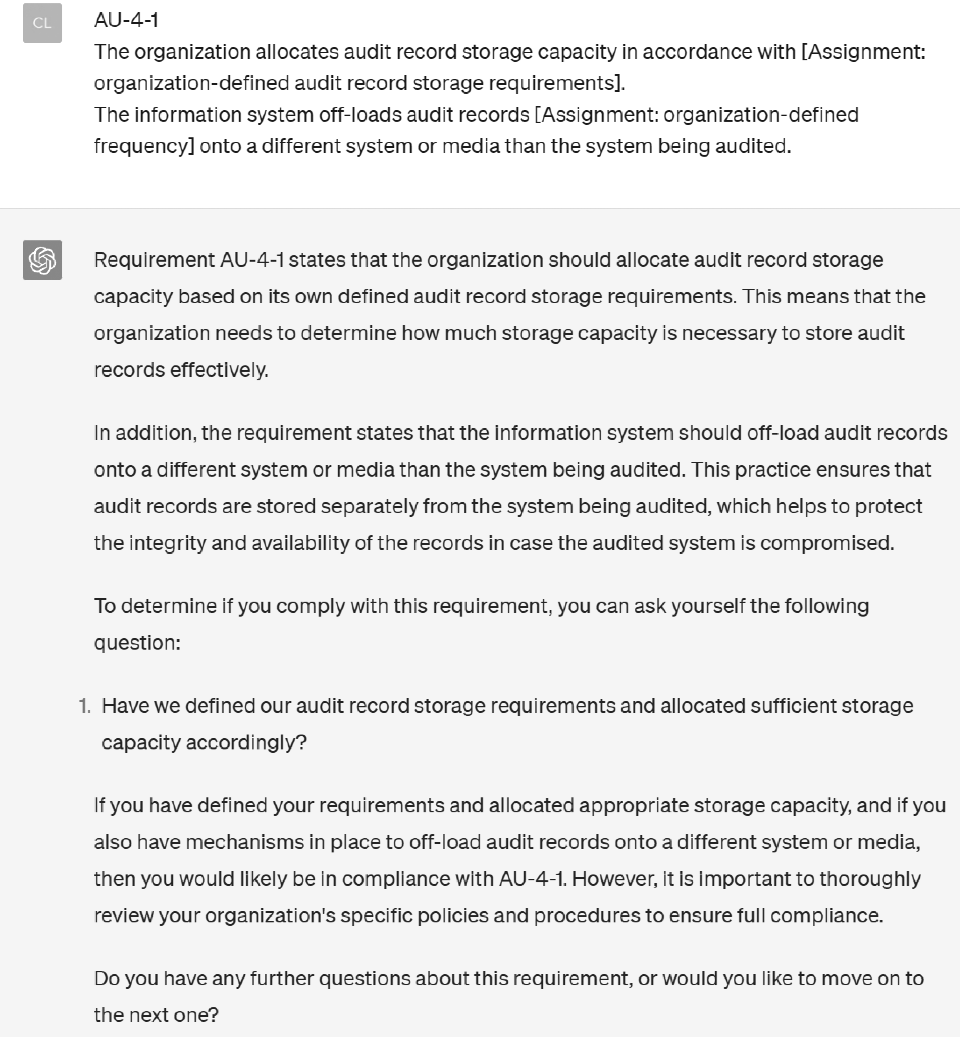


图 4.1 ChatGPT 对标准需求查询的响应示例

4.3.3 原理解释

将角色分配给 ChatGPT 时，已经为模型提供了特定的上下文或角色。这有助于模型生成针对特定角色的响应，从而生成更准确、更有相关性和更详细的内容。

在理解和检查网络安全标准合规性的上下文中，ChatGPT 可以使用其训练数据来解释标准要求，并将该要求分解为更简单、更易于理解的术语。然后形成一个问题，并帮助你确定是否符合该标准。

在整个过程中，你将与模型保持对话，根据你的需要，既可以深入研究某个特定需求，也可以转到下一个需求。

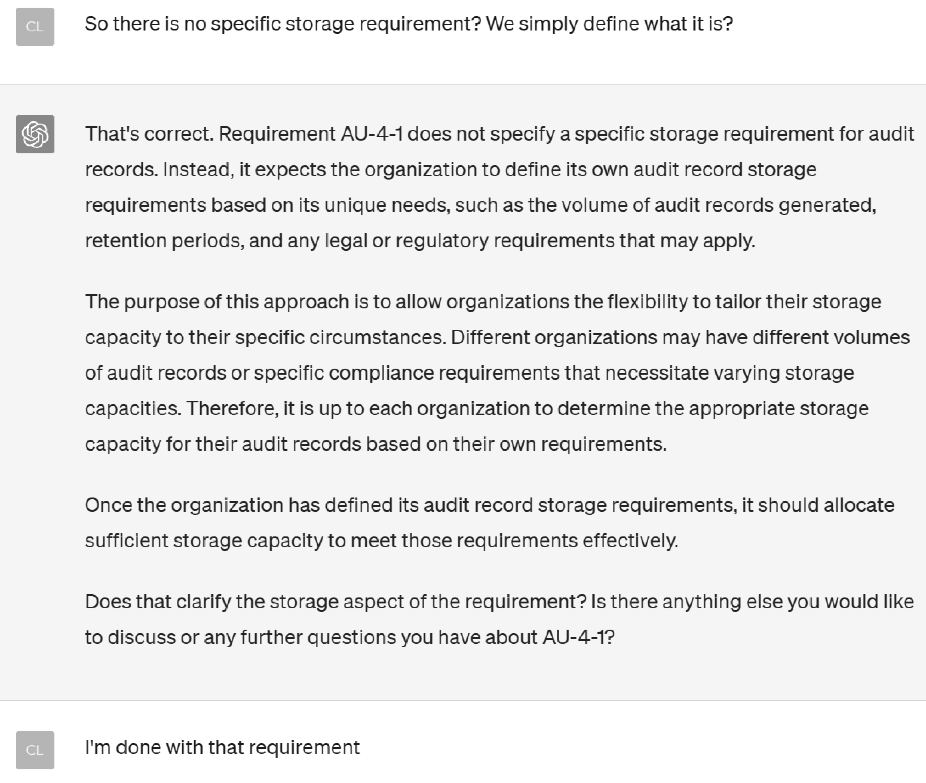


图 4.2 ChatGPT 对标准需求对话的响应示例

4.3.4 扩展知识

一旦熟悉了这个过程，即可将它扩展到不同行业的不同标准。

以下是一些需要考虑的额外要点：

● 使用 ChatGPT 作为培训辅助工具：你也可以将其用作教学工具，利用 ChatGPT 提供的简化解释来教育组织中的其他人了解不同网络安全标准的要求。使用该模型生成对复杂标准易于理解的解释，可以对更传统的培训形式进行有益的补充。

● 定期检查的重要性：定期使用 ChatGPT 来了解和检查是否符合网络安全标准是最有效的。网络安全环境变化很快，组织曾经遵守的要求可能会发生变化。定期检查可以帮助你的组织保持更新。

● 潜在的局限性：值得注意的是，虽然 ChatGPT 是一个强大的工具，但它确实有局限性。其响应基于截至 2021 年 9 月的训练数据。因此，对于最新的标准或自那时以来大幅更新的标准，其回应可能并不完全准确。这意味着使用最新版本的标准验证信息始终很重要。

 注意

下文将讨论提供更新文档作为知识库的更高级方法。

● 专业指导的重要性：虽然这种方法有助于理解网络安全标准的要求，但它不能取代专业的法律或网络安全指导。遵守这些标准往往会产生法律影响，因此专业建议至关重要。在确定你的组织是否符合任何网络安全标准时，请始终咨询专业人士。

● 反馈和迭代：与任何人工智能工具一样，你使用 ChatGPT 的次数越多，提供的反馈越多，它越能帮助你。反馈回路允许模型随着时间的推移调整并提供更适应你需求的响应。

4.4 创建风险评估流程

网络风险评估是组织风险管理战略的重要组成部分。这一过程包括：识别潜在威胁，评估这些威胁可能利用的漏洞，评估这种利用可能对组织产生的影响，以及建议采取适当的控制措施来减轻风险等。了解进行风险评估所涉及的步骤可以显著提高组织管理网络安全风险的能力。

本秘笈将指导你使用 Python 和 OpenAI API 创建网络风险评估流程。通过自动化风险评估流程，你可以简化工作流程，提高安全操作效率。这种方法还可以为进行风险评估提供标准化的格式，从而提高整个组织的一致性。

4.4.1 准备工作

在继续学习此秘笈之前，你需要做以下准备：

● Python：此秘笈与 Python 3.6 或更高版本兼容。

● 一个 OpenAI API 密钥。如果没有，则可以在注册后从 OpenAI 网站上获得。

● OpenAI Python 库。可使用以下命令安装：

**pip install openai**

● Python docx 库。用于创建 Word 文档。其安装命令如下：

**pip install python docx**

● Python tqdm 库。用于显示进度。其安装命令如下：

**pip install tqdm**

● Python threading 和 os 库，它们都是 Python 内置库。

● 熟悉 Python 编程和基本的网络安全概念。

4.4.2 实战操作

现在通过构建一个脚本来创建风险评估流程。该脚本将使用 OpenAI API 生成风险评估计划中每个部分的内容。脚本给 ChatGPT 分配的角色是一个专门从事 GRC 工作的网络安全专业人员，并提供风险评估过程的每个部分的详细叙述、背景和细节。

请按以下步骤操作：

（1）导入必要的库：

import openai

from openai import OpenAI

import os

from docx import Document

import threading

import time

from datetime import datetime

from tqdm import tqdm

此代码块可以为脚本导入所有必需的库，其中包括：用于与 OpenAI API 交互的 openai，用于环境变量的 os，用于创建 Word 文档的 Document（来自 docx），用于管理 API 调用期间的时间显示的 threading 和 time，用于对报告标记时间戳的 datetime，以及用于进度可视化的 tqdm。

（2）设置 OpenAI API 密钥：

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

此代码可以设置 OpenAI API 密钥，该密钥作为环境变量存储。此密钥用于验证我们的应用对 OpenAI API 的请求。

（3）确定评估报告的唯一标识符：

current\_datetime =

datetime.now().strftime('%Y-%m-%d\_%H-%M-%S')

assessment\_name =

f"Risk\_Assessment\_Plan\_{current\_datetime}"

上述代码使用当前日期和时间为每个评估报告创建一个唯一的名称，以确保不会覆盖以前的任何报告。

名称的格式为 Risk\_Assessment\_Plan\_{current\_datetime}，其中 current\_datetime 是运行脚本的确切日期和时间。

（4）定义风险评估大纲：

# Risk Assessment Outline

risk\_assessment\_outline = [

"Define Business Objectives",

"Asset Discovery/Identification",

"System Characterization/Classification",

"Network Diagrams and Data Flow Review",

"Risk Pre-Screening",

"Security Policy & Procedures Review",

"Cybersecurity Standards Selection and Gap

Assessment/Audit",

"Vulnerability Assessment",

"Threat Assessment",

"Attack Vector Assessment",

"Risk Scenario Creation (using the Mitre ATT&CK

Framework)",

"Validate Findings with Penetration Testing/Red

Teaming",

"Risk Analysis (Aggregate Findings & Calculate

Risk Scores)",

"Prioritize Risks",

"Assign Mitigation Methods and Tasks",

"Create Risk Report",

]

上述代码定义了风险评估的大纲。该大纲包含风险评估过程中要包括的所有部分的列表。

 提示

你可以修改流程步骤以包括你认为合适的部分，模型将填充你提供的任何部分的上下文。

（5）使用 OpenAI API 实现生成部分（节）内容的函数：

def generate\_section\_content(section: str) -> str:

# Define the conversation messages

messages = [

{

"role": "system",

"content": 'You are a cybersecurity

professional specializing in

governance, risk, and compliance (GRC)

with more than 25 years of

experience.'

},

{

"role": "user",

"content": f'You are

currently writing a cyber risk

assessment policy. Write the

narrative, context, and details for

the following section (and only

this section): {section}. Use as much

detail and explanation as possible. Do

not write anything that should go in

another section of the policy.'

},

]

# Call the OpenAI API

client = OpenAI()

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

# Return the generated text

Return

response.choices[0].message.content.strip()

此函数将风险评估大纲中某个部分的标题作为输入，并使用 OpenAI API 为该部分生成详细内容。

（6）实现将 Markdown 文本转换为 Word 文档的函数：

def markdown\_to\_docx(markdown\_text: str, output\_file: str):

document = Document()

# Iterate through the lines of the markdown text

for line in markdown\_text.split('\n'):

# Add headings based on the markdown heading

levels

if line.startswith('# '):

document.add\_heading(line[2:], level=1)

elif line.startswith('## '):

document.add\_heading(line[3:], level=2)

elif line.startswith('### '):

document.add\_heading(line[4:], level=3)

elif line.startswith('#### '):

document.add\_heading(line[5:], level=4)

# Add paragraphs for other text

else:

document.add\_paragraph(line)

# Save the Word document

document.save(output\_file)

此函数将为每个部分生成的 Markdown 文本和所需的输出文件名作为输入，并创建具有相同内容的 Word 文档。

（7）实现一个函数显示在等待 API 调用时经过的时间：

def display\_elapsed\_time():

start\_time = time.time()

while not api\_call\_completed:

elapsed\_time = time.time() - start\_time

print(f"\rElapsed time: {elapsed\_time:.2f}

seconds", end="")

time.sleep(1)

此函数负责显示在等待 API 调用完成时经过的时间。这对于跟踪过程所花费的时间非常有用。

（8）启动生成报告的过程：

api\_call\_completed = False

elapsed\_time\_thread =

threading.Thread(target=display\_elapsed\_time)

elapsed\_time\_thread.start()

在上述代码中，启动了一个单独的线程来显示经过的时间。这与进行 API 调用的主进程同时运行。

（9）遍历风险评估大纲中的每个部分，生成部分（节）内容，并将其附加到报告：

# Generate the report using the OpenAI API

report = []

pbar = tqdm(total=len(risk\_assessment\_outline),

desc="Generating sections")

for section in risk\_assessment\_outline:

try:

# Generate the section content

content = generate\_section\_content(section)

# Append the section content to the report

report.append(f"## {section}\n{content}")

except Exception as e:

print(f"\nAn error occurred during the API

call: {e}")

exit()

pbar.update(1)

此代码块将遍历风险评估大纲中的每一部分，使用 OpenAI API 生成部分（节）的内容，并将生成的内容附加到报告。

（10）生成所有部分的内容后，结束进度并显示经过的时间：

api\_call\_completed = True

elapsed\_time\_thread.join()

pbar.close()

api\_call\_completed 变量设置为 True，表示所有 API 调用都已完成。然后，我们停止运行时间显示线程，并关闭进度条以表示进程已结束。

（11）将生成的报告保存为 Word 文档：

# Save the report as a Word document

docx\_output\_file = f"{assessment\_name}\_report.docx"

# Handle exceptions during the report generation

try:

markdown\_to\_docx('\n'.join(report),

docx\_output\_file)

print("\nReport generated successfully!")

except Exception as e:

print(f"\nAn error occurred during the report

generation: {e}")

在最后一步中，使用已生成的报告（Markdown 格式）和所需的输出文件名作为参数调用 markdown\_to\_docx 函数，以创建一个 Word 文档。

文件名包括一个时间戳，以确保其唯一性。此进程被封装在 try-except 块中，用于处理此转换过程中可能发生的任何异常。如果成功，会打印出一条成功的消息；如果发生错误，则将打印异常以帮助进行故障排除。

完整脚本如下：

import openai

from openai import OpenAI

import os

from docx import Document

import threading

import time

from datetime import datetime

from tqdm import tqdm

# Set up the OpenAI API

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

current\_datetime = datetime.now()

.strftime('%Y-%m-%d\_%H-%M-%S')

assessment\_name =

f"Risk\_Assessment\_Plan\_{current\_datetime}"

# Risk Assessment Outline

risk\_assessment\_outline = [

"Define Business Objectives",

"Asset Discovery/Identification",

"System Characterization/Classification",

"Network Diagrams and Data Flow Review",

"Risk Pre-Screening",

"Security Policy & Procedures Review",

"Cybersecurity Standards Selection and Gap

Assessment/Audit",

"Vulnerability Assessment",

"Threat Assessment",

"Attack Vector Assessment",

"Risk Scenario Creation (using the Mitre ATT&CK

Framework)",

"Validate Findings with Penetration Testing/Red

Teaming",

"Risk Analysis (Aggregate Findings & Calculate Risk

Scores)",

"Prioritize Risks",

"Assign Mitigation Methods and Tasks",

"Create Risk Report",

]

# Function to generate a section content using the OpenAI

API

def generate\_section\_content(section: str) -> str:

# Define the conversation messages

messages = [

{

"role": "system",

"content": 'You are a cybersecurity

professional specializing in governance,

risk, and compliance (GRC) with more than

25 years of experience.'

},

{

"role": "user",

"content": f'You are currently writing a cyber

risk assessment policy. Write the

narrative, context, and details for the

following section (and only this section):

{section}. Use as much detail and

explanation as possible.

Do not write anything that should go in

another section of the policy.'

},

]

# Call the OpenAI API

client = OpenAI()

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

# Return the generated text

return response['choices'][0]['message']['content']

.strip()

# Function to convert markdown text to a Word document

def markdown\_to\_docx(markdown\_text: str, output\_file: str):

document = Document()

# Iterate through the lines of the markdown text

for line in markdown\_text.split('\n'):

# Add headings based on the markdown heading levels

if line.startswith('# '):

document.add\_heading(line[2:], level=1)

elif line.startswith('## '):

document.add\_heading(line[3:], level=2)

elif line.startswith('### '):

document.add\_heading(line[4:], level=3)

elif line.startswith('#### '):

document.add\_heading(line[5:], level=4)

# Add paragraphs for other text

else:

document.add\_paragraph(line)

# Save the Word document

document.save(output\_file)

# Function to display elapsed time while waiting for the

API call

def display\_elapsed\_time():

start\_time = time.time()

while not api\_call\_completed:

elapsed\_time = time.time() - start\_time

print(f"\rElapsed time: {elapsed\_time:.2f}

seconds", end="")

time.sleep(1)

api\_call\_completed = False

elapsed\_time\_thread =

threading.Thread(target=display\_elapsed\_time)

elapsed\_time\_thread.start()

# Generate the report using the OpenAI API

report = []

pbar = tqdm(total=len(risk\_assessment\_outline),

desc="Generating sections")

for section in risk\_assessment\_outline:

try:

# Generate the section content

content = generate\_section\_content(section)

# Append the section content to the report

report.append(f"## {section}\n{content}")

except Exception as e:

print(f"\nAn error occurred during the API call:

{e}")

api\_call\_completed = True

exit()

pbar.update(1)

api\_call\_completed = True

elapsed\_time\_thread.join()

pbar.close()

# Save the report as a Word document

docx\_output\_file = f"{assessment\_name}\_report.docx"

# Handle exceptions during the report generation

try:

markdown\_to\_docx('\n'.join(report), docx\_output\_file)

print("\nReport generated successfully!")

except Exception as e:

print(f"\nAn error occurred during the report

generation: {e}")

接下来，让我们看看它的工作原理。

4.4.3 原理解释

本秘笈中的 Python 脚本实际上是通过与 OpenAI API 交互来为风险评估过程的每个部分（节）生成详细内容。这些内容是通过模拟用户和系统（ChatGPT）之间的对话生成的，系统在其中扮演的是网络安全专业人员的角色。用户提供给 API 的对话消息描述了上下文，ChatGPT 基于该上下文生成全面的响应。

在 OpenAI 聊天模型中，提供了一个消息列表，每条消息都有一个角色和内容。角色可以是 system（系统）、user（用户）或 assistant（助手）。system 角色通常用于设置 assistant 的行为，user 角色用于指导 assistant。

在本示例脚本中，首先使用了以下消息来设置系统角色：

'You are a cybersecurity professional specializing in governance, risk, and compliance (GRC) with more than 25 years of experience.'

其含义是：你是一名拥有 25 年以上经验的网络安全专业人员，专门从事治理、风险和合规性（GRC）工作。

这是为了向模型提供背景信息，使其将自己定位为经验丰富的网络安全专业人员。模型将使用这些上下文信息来生成适用于特定场景的响应。

用户角色的消息如下：

'You are currently writing a cyber risk assessment

policy. Write the narrative, context, and details for the following

section (and only this section): {section}. Use as much detail and

explanation as possible. Do not write anything that should go in

another section of the policy.'

其含义是：你当前正在编写网络风险评估策略。请写出以下部分（仅此部分）的叙述、上下文和细节：｛section｝。应使用尽可能多的细节和解释，不要写任何应该写在策略另一部分的内容。

该消息是用户给模型的特定提示。此提示将引导模型为风险评估策略的特定部分生成详细说明。它指示模型聚焦于当前部分，不要偏离到其他部分的细节。通过这些提示，我们可以确保生成的内容具有相关性和准确性，并遵循风险评估流程的结构。

简而言之，system 角色设定了 assistant 的背景和专业知识，而 user 角色则为 assistant 提供了一项指导任务。这种方法有助于从人工智能中获得结构化且相关的内容。

该脚本结构化地处理风险评估过程的每个部分，对每个部分进行单独的 API 调用。它利用多线程的优势来显示处理 API 调用时经过的时间，从而为用户提供进度条。

为每个部分生成的内容都将附加到 Markdown 格式的报告中，然后使用 Python docx 库将其转换为 Word 文档。这将创建一个结构良好、详细的风险评估计划，可作为在组织中进行风险评估的起点。

4.4.4 扩展知识

此秘笈创建的风险评估过程是灵活的，你也可以尝试生成自己的风险评估流程，例如，使用 ChatGPT 编写不同部分的内容，然后将这些大纲部分插入脚本，这将使你能够创建一个适合你的组织的特定需求和风险状况的风险评估流程。

请记住，最好的风险评估过程就是根据反馈和新见解不断更新和改进的过程。

4.5 风险排序和优先级

本秘笈将利用 ChatGPT 的功能，根据给定的数据对网络安全风险进行优先级排序。网络安全风险的优先级排序是一项至关重要的任务，有助于组织将资源集中在最重要的地方。通过使用 ChatGPT，你可以使此任务更加易于管理和客观。

在给定的场景中，我们有一个数据集，其中包括不同资产或系统的一系列风险相关因素。这些因素包括资产类型、关键性评级、所服务的业务功能、攻击面的大小和评级、攻击向量评级以及现有的缓解措施和补救措施等。

ChatGPT 将帮助我们根据这些数据创建评分算法，以确定风险的优先级。根据评分算法计算的最高优先级风险将列在新表格的顶部。我们将使用示例数据指导你完成整个过程，但将来你也可以将相同的过程应用于自己的数据。

4.5.1 准备工作

登录你的 OpenAI 账户，确保你可以访问 ChatGPT 界面。

此外，你还需要一个数据集，其中包含系统及其相关漏洞的列表，以及和风险相关的数据。下文还将提供更多有关数据的说明。

如果你目前没有可用的数据集，则可以使用本秘笈中提供的数据集，其下载网址如下：

https://github.com/PacktPublishing/ChatGPT-for-Cybersecurity-Cookbook

4.5.2 实战操作

要对风险的优先级进行排序，可以向 ChatGPT 发送一个详细的提示。提示应明确说明任务并提供必要的上下文和数据。

 提示

你可以提供所需的任何系统数据，只要这些数据是分隔或描述性的，并且具有表示系统和漏洞的风险级别、严重性、价值等的标题名称和可识别值，ChatGPT 可以使用这些数据来创建适当的算法。

请按以下步骤操作：

（1）通过输入以下提示来建立系统角色：

You are a cybersecurity professional with 25 years of

experience.

（2）指示 ChatGPT 使用以下提示根据你的数据创建评分算法：

Based on the following dataset, categories, and values, create

a suitable risk scoring algorithm to help me prioritize the

risks and mitigation efforts. Provide me with the calculation

algorithm and then create a new table using the same columns,

but now ordered by highest priority to lowest (highest being on

top) and with a new column all the way to the left containing

the row number.

Data:

Asset/System Type Criticality Rating Business

Function Attack Surface Size Attack Surface

Rating Attack Vector Rating Mitigations and Remediations

Web Server

1 High Sales 120 Critical High Firewall

updates, SSL/TLS upgrades

Email

Server High Communication 80 High High Spam

filter updates, User training

File Server Medium HR 30 Medium Medium Apply

software patches, Improve password policy

Print Server Low All 15 Low Low Apply firmware

updates

Database Server

1 High Sales 200 Critical High Update DB

software, Enforce strong access control

Workstation

1 Low Engineering 10 Low Low Install

Antivirus, Apply OS patches

CRM

Software High Sales 50 Medium Medium Update

CRM software, Implement 2FA

ERP System High All 150 Critical High Update

ERP software, Implement access control

IoT Device

1 Low Maintenance 20 Medium Low Apply firmware

updates, Change default passwords

Web Server

2 Medium Marketing 60 Medium Medium SSL/TLS

upgrades, Implement WAF

Virtual Machine

1 Low Development 20 Low Low Apply OS patches,

Enforce strong access control

Networking

Switch High All 30 Medium High Firmware

updates, Change default credentials

Mobile Device

1 Medium Sales 25 Medium Medium Implement MDM,

Enforce device encryption

Firewall High All 70 High High Firmware

updates, Tighten rule sets

Cloud Storage High All 100 High High Apply

cloud security best practices, Implement 2FA

VOIP Server Medium All 45 Medium Medium Apply

patches, Implement strong access control

Payment

Gateway High Sales 180 Critical High Implement

PCI DSS, SSL/TLS upgrades

Router High All 30 Medium High Firmware

updates, Change default credentials

WiFi AP Medium All 40 Medium Medium Firmware

updates, Implement strong WiFi encryption

SAN Storage High All 60 High High Update SAN

software, Implement access control

图 4.3 显示了 ChatGPT 创建的评分算法输出：

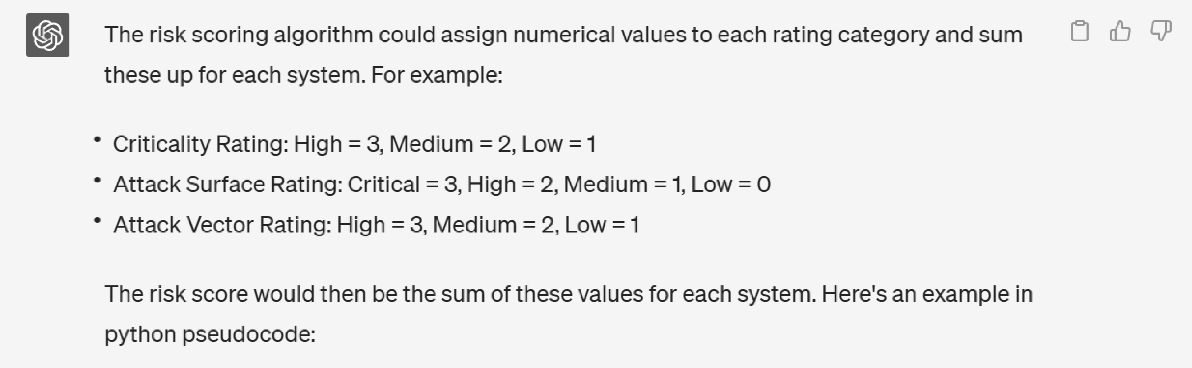


图 4.3 评分算法输出示例

以下代码是 ChatGPT 生成的示例评分算法输出：

def calculate\_risk\_score(criticality\_rating,

attack\_surface\_rating, attack\_vector\_rating):

ratings = {"High": 3, "Medium": 2, "Low": 1}

return ratings[criticality\_rating] +

ratings[attack\_surface\_rating] +

ratings[attack\_vector\_rating]

图 4.4 显示了来自 ChatGPT 的格式化输出示例，其中列出了各种网络安全资产，并根据算法创建的风险分数进行优先级排序：

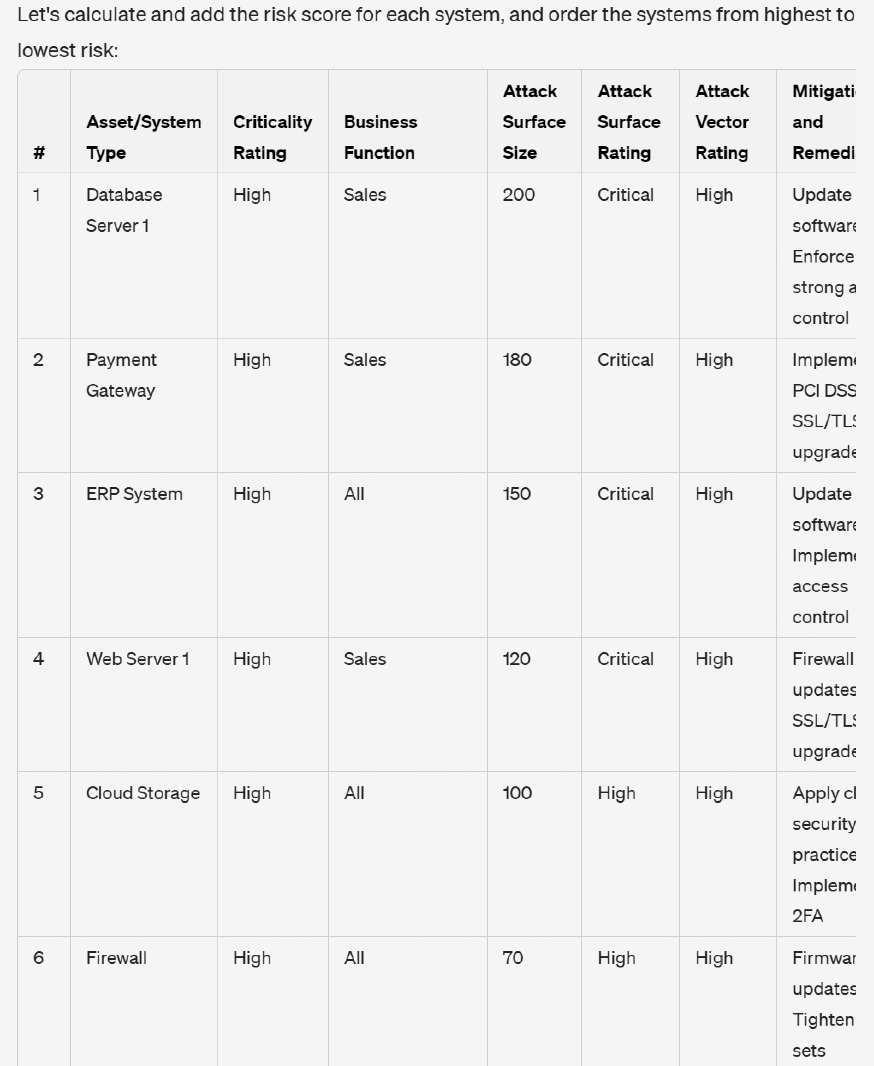


图 4.4 优先级输出示例

 提示

提示中提供的数据以制表符分隔。

 提示

本秘笈中使用的示例数据是通过以下提示生成的：

"Generate a table of sample data I will be using for a

hypothetical risk assessment example. The table should

be at least 20 rows and contain the following columns:

Asset/System Type, Criticality Rating, Business Function,

Attack Surface Size (a value that is derived from number of

vulnerabilities found on the system), Attack Surface

Rating (a value that is derived by calculating the

number of high and critical severity ratings compared to

the total attack surface),Attack Vector Rating (a value

that is derived by the number of other systems that have

access to this system, with internet facing being

the automatic highest number), list of mitigations

and remediations needed for this system (this would normally

be derived by the vulnerability scan recommendations based

on the findings but for this test/sample data, just make some

hypothetical data up.)"

4.5.3 原理解释

ChatGPT 基于一种被称为 Transformer 的机器学习模型，具体来说是一种称为生成式预训练 Transformer（generative pretrained Transformer，GPT）的变体。该模型已经在各种网络文本上进行了训练，学习了其中的语言模式和事实信息，并从庞大的语料库中获得一定的推理能力。

当被赋予创建风险评分算法的任务时，ChatGPT 并没有利用对网络安全或风险管理的固有理解。相反，它利用了在训练阶段学到的模式。在训练过程中，它很可能遇到过与风险评分算法、风险优先级和网络安全相关的文本。通过识别训练数据中此类信息的结构和上下文，它可以在获得提示时生成相关且连贯的响应。

创建风险评分算法时，ChatGPT 将首先了解数据中的各种因素，如 Criticality Rating（关键性评级）、Business Function（业务功能）、Attack Surface Size（攻击面大小）、Attack Surface Rating（攻击面评级）、Attack Vector Rating（攻击向量评级）以及 Mitigations and Remediations（缓解和补救措施）。它认识到这些因素在确定与每项资产相关的整体风险方面很重要。然后，ChatGPT 制定了一种算法，将这些因素考虑在内，根据它们在整体风险评估中的重要性，为每个因素分配不同的权重和分数。

然后，模型将生成的算法应用于数据，对每个风险进行评分，创建一个按这些评分排序的新表。这种排序过程有助于确定风险优先级—分数较高的风险被认为更关键，并列在表的顶部。

ChatGPT 令人印象深刻的一点是，虽然它并不能真正理解人类意义上的网络安全或风险评估，但它可以根据所学的模式令人信服地模仿这种理解。它能够基于这些模式生成创造性和连贯的文本，这使它成为一种可用于各种任务的通用工具，自然，这里所谓的各种任务也包括本秘笈中的生成风险评分算法。

4.5.4 扩展知识

此方法受 ChatGPT 的 token 限制。由于该限制，你只能粘贴有限的数据。当然，本书后面将告诉你如何使用更高级的技巧绕过这一限制。

 提示

不同的模型有不同的 token 限制。如果你是 OpenAI Plus 的订阅用户，则可以在 GPT-3.5 和 GPT-4 模型之间进行选择。GPT-4 的 token 限制是 GPT-3.5 的2倍。

此外，如果你使用 OpenAI Playground 而不是 ChatGPT 用户界面，则可以使用新的 gpt-3.5-turbo-16k 模型，它的 token 限制是 GPT-3.5 的 4 倍。

4.6 构建风险评估报告

网络安全涉及管理和减轻风险，这一过程的一个重要部分是创建详细的风险评估报告。此类报告不仅需要记录已确定的风险、漏洞和威胁，还需要阐明为解决这些问题所采取的步骤，以促进与各利益攸关方的沟通。自动创建风险评估报告可以节省大量时间，并确保报告之间的一致性。

本秘笈将创建一个 Python 脚本，该脚本可使用 OpenAI 的 ChatGPT 自动生成网络风险评估报告。我们将使用由用户提供的数据—虽然本节仍然以 4.5 节“风险排序和优先级”秘笈中使用的数据为例，但是，脚本和提示已被设计为可处理用户提供的任何相关数据。

到本秘笈结束时，你将能够使用 Python、ChatGPT 和你自己的数据生成详细而一致的风险评估报告。

4.6.1 准备工作

在继续学习此秘笈之前，你需要做以下准备：

● Python。

● OpenAI Python 库。其安装命令如下：

**pip install openai**

● python-docx 库。用于创建 Word 文档。其安装命令如下：

**pip install python-docx**

● Python tqdm 库。用于显示进度。其安装命令如下：

**pip install tqdm**

● OpenAI API 密钥。

4.6.2 实战操作

在开始实际操作之前，请记住，你需要在 systemdata.txt 文件中提供系统数据。这些数据可以是任何形式，只要它们是被分隔的或描述性的，并包含可辨别的值，这些值表示系统和漏洞的风险水平、严重性和价值等。ChatGPT 将使用这些信息创建适当的算法并生成上下文准确的报告部分。

请按以下步骤操作：

（1）导入所需的库：

import openai

from openai import OpenAI

import os

from docx import Document

import threading

import time

from datetime import datetime

from tqdm import tqdm

这些库都是脚本正确运行所必需的。其中，openai 用于与 OpenAI API 交互，os 用于访问环境变量，Document（来自 docx）用于创建 Word 文档，threading 和 time 用于多线程和跟踪运行经过的时间，datetime 用于为每次运行生成唯一的文件名，而 tqdm 用于在控制台中显示进度条。

（2）设置 OpenAI API 密钥并生成评估名称：

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

current\_datetime = datetime.now()

.strftime('%Y-%m-%d\_%H-%M-%S')

assessment\_name =

f"Risk\_Assessment\_Plan\_{current\_datetime}"

上述代码可从环境变量中读取 OpenAI API 密钥，并使用当前日期和时间为风险评估报告创建唯一的文件名。

（3）创建风险评估报告大纲：

risk\_assessment\_outline = [

"Executive Summary",

"Introduction",

# More sections...

]

这是风险评估报告的结构，用于指导人工智能模型为每个部分生成内容。

（4）定义生成各部分内容的函数：

def generate\_section\_content(section: str,

system\_data: str) -> str:

messages = [

{

"role": "system",

"content": 'You are a cybersecurity

professional...'

},

{

"role": "user",

"content": f'You are currently

writing a cyber risk assessment

report...{system\_data}'

},

]

# Call the OpenAI API

client = OpenAI()

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

Return

response.choices[0].message.content.strip()

此函数构造了一个对话提示，将其发送到 OpenAI API，并检索模型的响应。它接受部分（节）的名称和系统数据作为参数，并返回为指定部分生成的内容。

（5）定义将 Markdown 文本转换为 Word 文档的函数：

def markdown\_to\_docx(markdown\_text: str, output\_file: str):

document = Document()

# Parsing and conversion logic...

document.save(output\_file)

此函数接收 Markdown 格式的文本和一个文件路径，可根据 Markdown 内容创建 Word 文档，并将文档保存到指定的文件路径。

（6）定义显示经过的时间的函数：

def display\_elapsed\_time():

start\_time = time.time()

while not api\_call\_completed:

elapsed\_time = time.time() - start\_time

print(f"\rElapsed time: {elapsed\_time:.2f}

seconds", end="")

time.sleep(1)

此函数用于在等待 API 调用完成时，在控制台中显示运行时间。它被实现为一个单独的线程，以允许主线程继续执行脚本的其余部分。

（7）读取系统数据并启动运行时间的线程：

with open("systemdata.txt") as file:

system\_data = file.read()

api\_call\_completed = False

elapsed\_time\_thread =

threading.Thread(target=display\_elapsed\_time)

elapsed\_time\_thread.start()

该脚本将从文本文件中读取系统数据，并启动一个新线程以在控制台中显示经过的时间。

（8）使用 OpenAI API 生成报告：

report = []

pbar = tqdm(total=len(risk\_assessment\_outline),

desc="Generating sections")

for section in risk\_assessment\_outline:

try:

content = generate\_section\_content(section,

system\_data)

report.append(f"## {section}\n{content}")

except Exception as e:

print(f"\nAn error occurred during the API

call: {e}")

api\_call\_completed = True

exit()

pbar.update(1)

api\_call\_completed = True

elapsed\_time\_thread.join()

pbar.close()

该脚本将创建一个进度条，在风险评估报告大纲的各个部分（节）循环迭代，使用 OpenAI API 为每个部分生成详细内容，并将内容附加至报告。在完成之后，它将停止运行时间线程并关闭进度条。

（9）将报告另存为 Word 文档：

docx\_output\_file = f"{assessment\_name}\_report.docx"

try:

markdown\_to\_docx('\n'.join(report),

docx\_output\_file)

print("\nReport generated successfully!")

except Exception as e:

print(f"\nAn error occurred during the report

generation: {e}")

最后，脚本将生成的报告从 Markdown 格式转换为 Word 文档并保存该文档。如果在此过程中抛出异常，则会捕获该异常并将消息打印到控制台。

完整脚本如下：

import openai

from openai import OpenAI

import os

from docx import Document

import threading

import time

from datetime import datetime

from tqdm import tqdm

# Set up the OpenAI API

openai.api\_key = os.getenv("OPENAI\_API\_KEY")

current\_datetime = datetime.now()

.strftime('%Y-%m-%d\_%H-%M-%S')

assessment\_name =

f"Risk\_Assessment\_Plan\_{current\_datetime}"

# Cyber Risk Assessment Report Outline

risk\_assessment\_outline = [

"Executive Summary",

"Introduction",

"Asset Discovery/Identification",

"System Characterization/Classification",

"Network Diagrams and Data Flow Review",

"Risk Pre-Screening",

"Security Policy & Procedures Review",

"Cybersecurity Standards Selection and Gap

Assessment/Audit",

"Vulnerability Assessment",

"Threat Assessment",

"Attack Vector Assessment",

"Risk Scenario Creation (using the Mitre ATT&CK

Framework)",

"Validate Findings with Penetration Testing/Red

Teaming",

"Risk Analysis (Aggregate Findings & Calculate Risk

Scores)",

"Prioritize Risks",

"Assign Mitigation Methods and Tasks",

"Conclusion and Recommendations",

"Appendix",

]

# Function to generate a section content using the OpenAI

API

def generate\_section\_content(section: str, system\_data:

str) -> str:

# Define the conversation messages

messages = [

{

"role": "system",

"content": 'You are a cybersecurity

professional specializing in governance,

risk, and compliance (GRC) with more than

25 years of experience.'

},

{

"role": "user",

"content": f'You are currently writing a

cyber risk assessment report. Write the

context/details for the following section

(and only this section): {section}, based

on the context specific that section, the

process that was followed, and the

resulting system data provided below. In

the absense of user provided context or

information about the process followed,

provide placeholder context that aligns

with industry standard context for that

section. Use as much detail and explanation

as possible. Do not write

anything that should go in another section

of the policy.\n\n{system\_data}'

},

]

# Call the OpenAI API

client = OpenAI()

response = client.chat.completions.create(

model="gpt-3.5-turbo",

messages=messages,

max\_tokens=2048,

n=1,

stop=None,

temperature=0.7,

)

# Return the generated text

return response.choices[0].message.content.strip()

# Function to convert markdown text to a Word document

def markdown\_to\_docx(markdown\_text: str, output\_file: str):

document = Document()

# Iterate through the lines of the markdown text

for line in markdown\_text.split('\n'):

# Add headings based on the markdown heading levels

if line.startswith('# '):

document.add\_heading(line[2:], level=1)

elif line.startswith('## '):

document.add\_heading(line[3:], level=2)

elif line.startswith('### '):

document.add\_heading(line[4:], level=3)

elif line.startswith('#### '):

document.add\_heading(line[5:], level=4)

# Add paragraphs for other text

else:

document.add\_paragraph(line)

# Save the Word document

document.save(output\_file)

# Function to display elapsed time while waiting for the

API call

def display\_elapsed\_time():

start\_time = time.time()

while not api\_call\_completed:

elapsed\_time = time.time() - start\_time

print(f"\rElapsed time: {elapsed\_time:.2f}

seconds", end="")

time.sleep(1)

# Read system data from the file

with open("systemdata.txt") as file:

system\_data = file.read()

api\_call\_completed = False

elapsed\_time\_thread =

threading.Thread(target=display\_elapsed\_time)

elapsed\_time\_thread.start()

# Generate the report using the OpenAI API

report = []

pbar = tqdm(total=len(risk\_assessment\_outline),

desc="Generating sections")

for section in risk\_assessment\_outline:

try:

# Generate the section content

content = generate\_section\_content(section,

system\_data)

# Append the section content to the report

report.append(f"## {section}\n{content}")

except Exception as e:

print(f"\nAn error occurred during the API call:

{e}")

exit()

pbar.update(1)

api\_call\_completed = True

elapsed\_time\_thread.join()

pbar.close()

# Save the report as a Word document

docx\_output\_file = f"{assessment\_name}\_report.docx"

# Handle exceptions during the report generation

try:

markdown\_to\_docx('\n'.join(report), docx\_output\_file)

print("\nReport generated successfully!")

except Exception as e:

print(f"\nAn error occurred during the report

generation: {e}")

接下来，让我们看看它的工作原理。

4.6.3 原理解释

该脚本的关键功能是根据系统数据和评估过程自动生成详细的风险评估报告。该脚本将流程划分为一系列定义的部分，并在每个部分中使用 OpenAI API 生成特定的详细内容。

从文件加载的系统数据为 gpt-3.5-turbo 模型提供了上下文，以生成每个部分的内容。我们创建了一个大纲，将风险评估报告分解为多个部分，每个部分代表风险评估过程中的一个阶段。这些部分与 4.4 节“创建风险评估流程”秘笈中概述的步骤基本相似。

我们使用了以下提示在脚本中构建一个报告模板提示：

You are a cybersecurity professional and CISO with more than 25 years

of experience. Create a detailed cyber risk assessment report outline

that would be in line with the following risk assessment process

outline:

1. Define Business Objectives

2. Asset Discovery/Identification

3. System Characterization/Classification

4. Network Diagrams and Data Flow Review

5. Risk Pre-Screening

6. Security Policy & Procedures Review

7. Cybersecurity Standards Selection and Gap Assessment/Audit

8. Vulnerability Assessment

9. Threat Assessment

10. Attack Vector Assessment

11. Risk Scenario Creation (using the Mitre ATT&CK Framework)

12. Validate Findings with Penetration Testing/Red Teaming

13. Risk Analysis (Aggregate Findings & Calculate Risk Scores)

14. Prioritize Risks

15. Assign Mitigation Methods and Tasks"

这种方法将指导模型生成与报告的每个部分相匹配的内容。

在报告大纲的每个部分中，脚本将调用 generate\_section\_content()函数。该函数向 OpenAI API 发送一条聊天消息，将模型的角色定义为一名经验丰富的网络安全专业人员，同时提示当前的任务（编写特定部分的详细内容）和提供的系统数据。模型将生成每个特定部分的详细内容，然后由函数返回并添加到 report 列表。

markdown\_to\_docx()函数的作用是将 report 列表中的 Markdown 文本转换为 Word 文档。它将迭代 Markdown 文本中的每一行，检查它是否以 Markdown 标题标记（如#、##等）开头，并相应地将其作为标题或段落添加到文档。

在生成所有部分的详细内容并将其附加到 report 列表后，该列表将合并成一个字符串，并使用 markdown\_to\_docx()函数转换为 Word 文档。

4.6.4 扩展知识

在本示例中，评估报告每个部分的上下文都是占位符文本，你也可以修改这些文本。为了简单起见，我们使用了这种方法，在后面的秘笈中，我们将展示更高级的技巧，以演示如何将实际的风险评估过程作为报告的真实上下文。

我们鼓励你尝试不同的评估流程大纲和数据集。了解如何调整提示和数据以获得最有效的结果，是利用如 gpt-3.5-turbo 和 gpt-4 等 AI 模型来满足你的需求的关键部分。

 注意

请记住，与前面的秘笈类似，此方法也受所选模型的 token 限制。gpt-3.5-turbo 模型的 token 限制为 4096，这限制了可以从系统数据文件传入的数据量。本书后面将探索一些高级技巧来绕过这一限制。有了这些技巧，你将能够处理更大的数据集并生成更全面的报告。

 提示

与本书中的大多数秘笈一样，本章中的秘笈使用了 gpt-3.5-turbo 模型，该基线设置是最具成本效益的模型。我们鼓励你尝试使用不同的模型，如 gpt-3.5-turbo、gpt-4 和最新发布的 gpt-3.5-turbo-16k，以找到最适合你需求的结果。