**2024-01-02 漏洞处理**

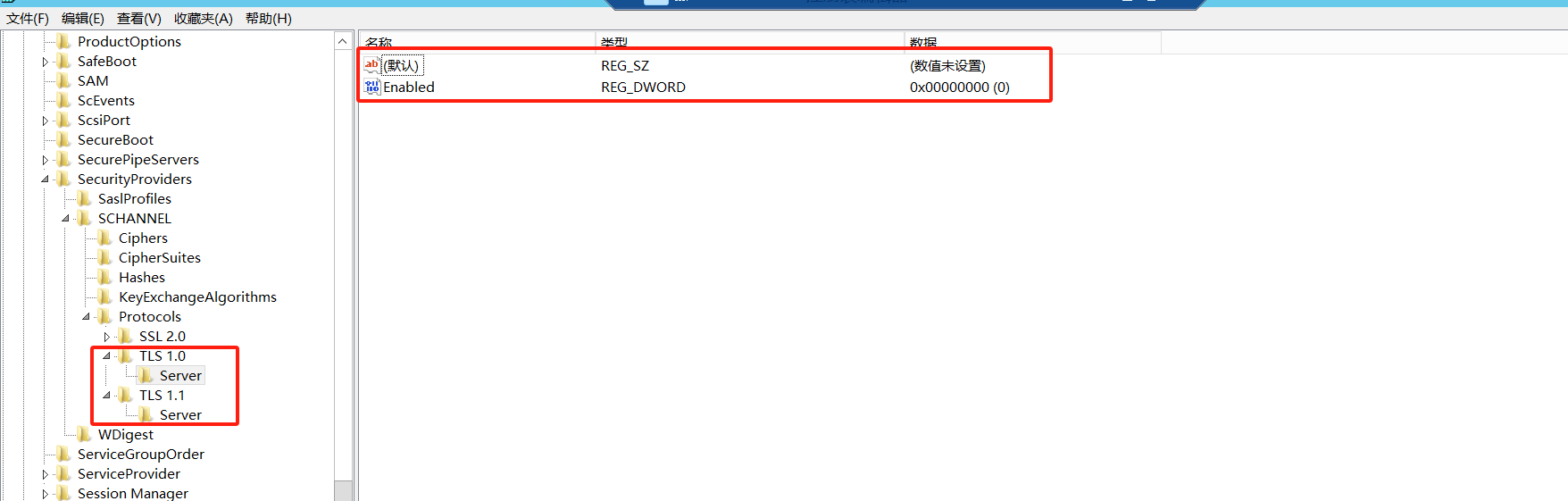
**实施模块：**

1. **TLS 1.0 已启用与TLS 1.1 已启用 设置禁用的方式**

1、按下Win + R键，在运行对话框中输入"regedit"，然后点击"确定"打开注册表编辑器。

2、航到以下路径 没有新增配置

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\Server



3、创建或修改注册表项：在"Server"键下，如果不存在名为"Enabled"的DWORD值，请右键点击"Server"键，选择"新建"，然后选择"DWORD (32位)值"。将新创建的值命名为"Enabled"。

设置注册表项值：双击"Enabled"值，将数值数据设置为0以禁用TLS 1.0

1. 配置appsettings配置如下配置

"MiddlewareSettings": { //中间件的配置

"IsCors": "1", //是否跨域资源共享：0不启动，1启动

"WithOrigins": "https://localhost:44315,https://localhost:5001", //WithOrigins方法中传递一个字符串数组，包含所有允许的网址 多个使用逗号隔开，网址不用以“/”结束

//这段内容是一个CSP（Content Security Policy）的配置，它是一种增强Web应用程序安全性的技术

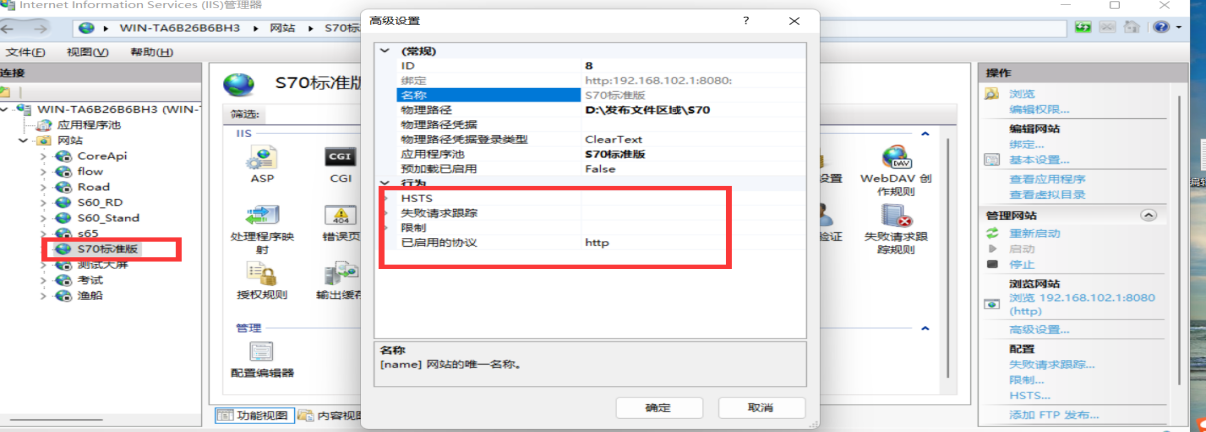
"CSP": "default-src 'self';connect-src \*; script-src 'self' 'unsafe-inline' 'unsafe-eval' ; style-src 'self' 'unsafe-inline';img-src \*;frame-src 'self' ;font-src 'self' data:; block-all-mixed-content; "

}

三、未实施 HTTP 严格传输安全 (HSTS)

IIS部署进行配置处理

部署网站右键→管理网站→高级设置



**开发模块**

1. **CORS（跨域资源共享）**
2. 需要配置两个参数，分别是

"MiddlewareSettings": { //中间件的配置

"IsCors": "1", //是否跨域资源共享：0不启动，1启动

"WithOrigins": "https://localhost:44315,https://localhost:5001", //WithOrigins方法中传递一个字符串数组，包含所有允许的网址 多个使用逗号隔开，网址不用以“/”结束

}

修改是Startup文件下，修改AddCores的服务

代码如下与截图

services.AddCors(options =>

{

options.AddPolicy("CorsPolicy", builder =>

{

string IsCors = Configuration.GetValue<string>("MiddlewareSettings:IsCors");

if (IsCors == "1")

{

string WithOrigins = Configuration.GetValue<string>("MiddlewareSettings:WithOrigins") ?? "";

builder.WithOrigins(WithOrigins.Split(','))

.AllowAnyMethod()

.AllowAnyHeader()

.AllowCredentials();

}

else

{

builder.SetIsOriginAllowed(\_ => true)

.AllowAnyMethod()

.AllowAnyHeader()

.AllowCredentials();

}

});

});



1. **所有中间件添加请求头如下**

1、**设置代码如下**

app.Use(async (context, next) =>

{

string CSP = Configuration.GetValue<string>("MiddlewareSettings:CSP") ?? "";

if (!string.IsNullOrWhiteSpace(CSP))

{

context.Response.Headers.Add("Content-Security-Policy", CSP);

}

context.Response.Headers.Add("Cache-Control", "no-cache, no-store, must-revalidate");

context.Response.Headers.Add("Pragma", "no-cache");

context.Response.Headers.Add("X-Frame-Options", "SAMEORIGIN");

await next();

});



**X-Frame-Options 报头缺失**

1. X-Frame-Options: 这是一个HTTP响应头部，用于控制页面是否可以被嵌入到<iframe>、<frame>或<object>中。
2. 配置说明

（1）只允许同源页面嵌入:

context.Response.Headers.Add("X-Frame-Options", "SAMEORIGIN");

1. 完全禁止任何外部页面嵌入:

context.Response.Headers.Add("X-Frame-Options", "DENY");

1. 只允许来自特定域的页面嵌入:

context.Response.Headers.Add("X-Frame-Options", "ALLOW-FROM [https://example.com");](https://example.com\");)

**可缓存敏感页面**

ASP.NET中用于配置HTTP响应头部信息的代码，用于控制页面的缓存行为。

1、context.Response.Headers.Add("Cache-Control", "no-cache, no-store, must-revalidate");

Cache-Control: 是一个HTTP头部字段，用于定义缓存行为。

no-cache: 指示浏览器或其他中间缓存代理，每次请求资源时都要重新验证其内容是否最新。简单地说，这意味着资源总是被视为未缓存，每次都需要重新请求服务器来确认内容是否更改。

no-store: 指示浏览器或其他中间缓存代理不得存储任何版本的资源。这意味着即使资源后来更改，也不会在缓存中保留任何旧版本。

must-revalidate: 如果中间缓存有一个已缓存的版本，它必须使用强比较（即请求原始响应头和实体标签）来验证响应是否仍然有效。

2、context.Response.Headers.Add("Pragma", "no-cache");

Pragma: 是一个较老的HTTP头部字段，用于控制页面缓存。

no-cache: 指示浏览器或其他中间缓存代理不要缓存页面内容。

**未实施内容安全策略 (CSP)**

1、需要配置一个参数如下

"MiddlewareSettings": { //中间件的配置

//这段内容是一个CSP（Content Security Policy）的配置，它是一种增强Web应用程序安全性的技术

"CSP": "default-src 'self';connect-src \*; script-src 'self' 'unsafe-inline' 'unsafe-eval' ; style-src 'self' 'unsafe-inline';img-src \*;frame-src 'self' ;font-src 'self' data:; block-all-mixed-content; "

}

这段内容是一个CSP（Content Security Policy）的配置，它是一种增强Web应用程序安全性的技术。CSP通过限制Web页面可以加载和执行的资源类型，来防止某些类型的攻击，例如跨站脚本攻击（XSS）。

下面是对这段配置的逐条解释：

default-src 'self';

这是默认的源，表示除非明确指定，否则所有资源都只能从同源URL加载。'self' 指的是当前域名。

connect-src \*;

这定义了哪些源可以用于创建网络连接。星号 (\*) 表示允许来自任何源的连接。这通常用于WebSockets和EventSource。

script-src 'self' 'unsafe-inline' 'unsafe-eval';

这定义了哪些源可以执行或注入脚本。'self' 允许从同源URL加载脚本，'unsafe-inline' 允许内联脚本（尽管不推荐），'unsafe-eval()' 允许使用 eval() 和其他动态代码执行函数。

style-src 'self' 'unsafe-inline';

这定义了哪些源可以加载样式表。'self' 允许从同源URL加载样式表，'unsafe-inline' 允许内联样式。

img-src \*;

这定义了哪些源可以加载图片。星号 (\*) 表示允许来自任何源的图片。

frame-src 'self';

这定义了哪些源可以用来嵌入页面（例如，通过 <frame> 或 <iframe>）。'self' 仅允许从同源URL嵌入页面。

font-src 'self' data:;

这定义了哪些源可以加载字体。'self' 允许从同源URL加载字体，而 data: 表示允许从data URL加载字体。

block-all-mixed-content;

这表示应阻止所有混合内容。混合内容是同时使用安全（HTTPS）和非安全（HTTP）内容的页面。阻止混合内容有助于增强页面的安全性。

1. **Microsoft IIS 版本披露**

代码如下

return Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.ConfigureKestrel(options =>

{

options.AllowSynchronousIO = true;

options.AddServerHeader = false;

});

webBuilder.UseKestrel(option =>

{

option.AddServerHeader = false;

});

webBuilder.UseStartup<Startup>();

});

