# BOLETIM DE SEGURANÇA

## Microsoft descobre vulnerabilidades críticas no Rockwell Automation PanelView Plus

heimdall
security research
A DIVISION OF ISH

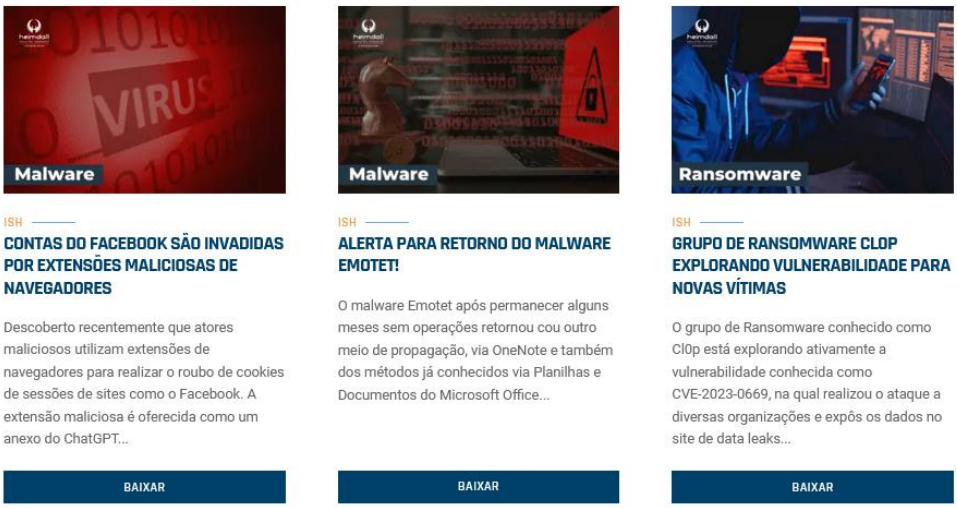
TLP: CLEAR

Receba alertas e informações sobre segurança cibernética e ameaças rapidamente, por meio do nosso **X**.

[Heimdall Security Research]


Acesse boletins diários sobre agentes de ameaças, *malwares*, indicadores de comprometimentos, TTPs e outras informações no *site* da ISH.

[Boletins de Segurança – Heimdall]



## SUMÁRIO

# LISTA DE FIGURAS

# 1 SUMÁRIO EXECUTIVO

A Microsoft identificou duas vulnerabilidades de segurança significativas no ***Rockwell Automation PanelView Plus***. Essas falhas podem ser exploradas por invasores remotos não autenticados para executar código arbitrário e desencadear uma condição de negação de serviço (DoS), representando um risco potencial para a segurança dos sistemas.

# 2 DETALHES DAS EXPLORAÇÕES

Os dispositivos PanelView Plus são terminais gráficos, também conhecidos como interfaces homem-máquina (HMI), amplamente utilizados no setor industrial. Essas vulnerabilidades podem ter um impacto significativo nas organizações que utilizam os dispositivos afetados, pois permitem aos invasores executarem códigos remotamente e interromper as operações.

A vulnerabilidade **CVE-2023-2071** é uma falha de validação de entrada imprópria que permite a invasores não autenticados executar código remoto por meio de pacotes maliciosos. Por outro lado, a vulnerabilidade **CVE-2023-29464** é uma falha de validação de entrada imprópria que permite a um agente de ameaça não autenticado ler dados da memória por meio de pacotes maliciosos e causar um DoS ao enviar um pacote maior que o tamanho do buffer.

Durante a análise, notou-se uma troca legítima de pacotes entre dois dispositivos utilizando o Common Industrial Protocol (CIP). Um dos dispositivos enviava uma solicitação contendo um caminho para um valor de registro chamado “ProductCode”, e o outro respondia com o que parecia ser o valor do código do produto. A ausência de criptografia e autenticação prévia na comunicação levantou preocupações, pois parecia envolver uma consulta remota de registro. Investigações adicionais revelaram que o dispositivo que fazia a solicitação era uma estação de trabalho de engenharia e o dispositivo que respondia era um HMI - mais especificamente, um PanelView Plus.

Suspeita-se que essa funcionalidade de consulta remota de registro poderia ser explorada para consultar chaves do sistema a fim de acessar informações confidenciais ou até mesmo obter controle remoto. Para confirmar essa suspeita, precisávamos encontrar o código responsável por essa funcionalidade. Como a comunicação entre os dois dispositivos era feita através do CIP, nosso primeiro passo foi compreender profundamente esse protocolo.

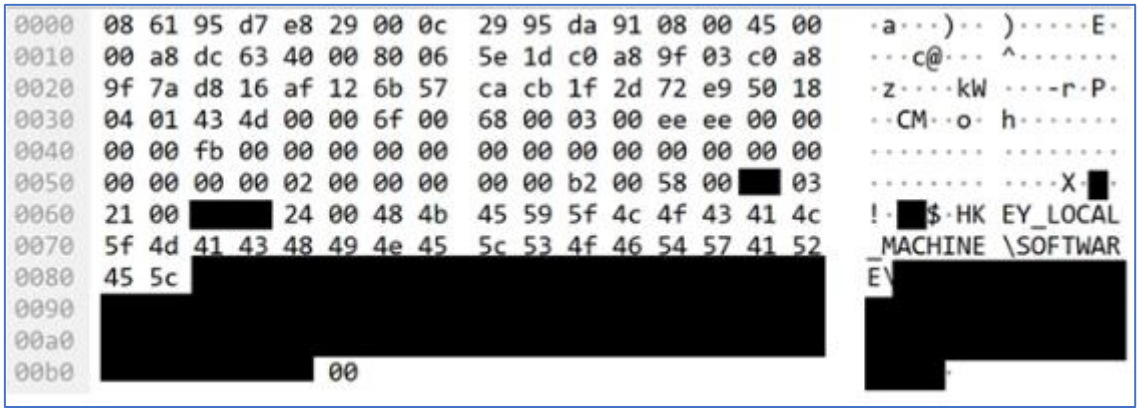



*Figura 1 – Análise do pacote em um editor hexadecimal.*

O IP é um protocolo industrial criado para aplicações de automação industrial. Esse protocolo é amplamente utilizado por diversos fornecedores no setor industrial, e a comunicação que observamos foi realizada por meio do Ethernet/IP - um protocolo que adapta o CIP ao Ethernet padrão. Segundo a documentação oficial do CIP, um nó CIP é modelado como uma coleção de Objetos. Uma Classe é um conjunto de Objetos que representam o mesmo tipo de componente do sistema. Uma Instância de Objeto é a representação real de um Objeto específico dentro de uma Classe.

Com base nesta descrição, podemos inferir que o CIP é um protocolo orientado a objetos, onde as mensagens são direcionadas a objetos específicos, identificados por seu ID de classe e ID de instância de objeto. Além disso, o termo "Service Code" é definido como um valor de identificação inteiro que denota uma solicitação de ação que pode ser direcionada a uma instância de objeto ou atributo de objeto específico. Portanto, ao enviar mensagens a um objeto, também devemos especificar um Código de serviço, que informa ao objeto qual ação ele deve executar.

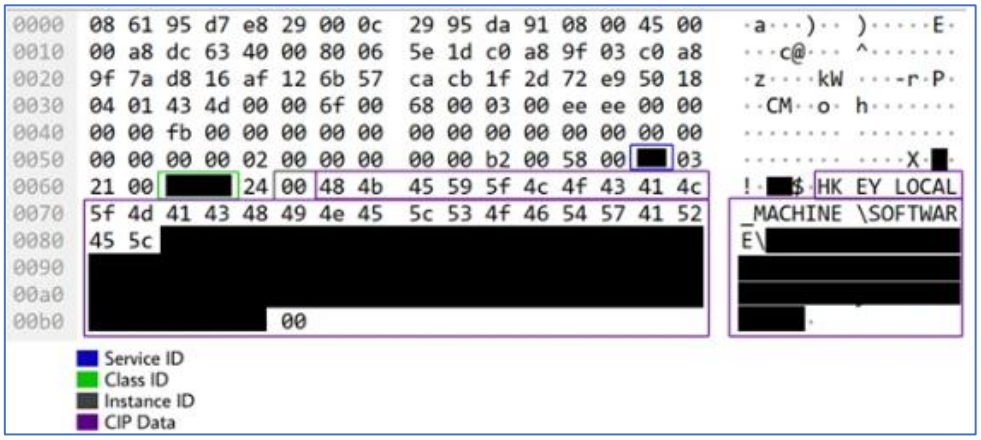



*Figura 2 – Especificação CIP que descreve IDs de classe e IDs de serviço.*

Ao analisar a captura de pacotes, notou-se que os valores de Service ID e Class ID eram específicos do fornecedor. Isso implica que, para compreender o significado desses IDs de Classe e Serviço e identificar o código responsável pela funcionalidade, é necessário examinar o firmware do HMI. De acordo com informações disponíveis pela Rockwell Automation, os HMIs PanelView Plus funcionam no sistema operacional Windows 10 IoT (ou Windows CE para versões mais antigas). Fomos capazes de extrair as DLLs e executáveis relacionados à Rockwell Automation do firmware mais recente. Existem várias DLLs que recebem diferentes IDs de Classe e processam suas solicitações, uma das quais é responsável por processar o ID de Classe que observamos na captura de pacotes.

```
ReturnCode = ParseRegistryPath((const char *)CipData, (int *)&hKey, (int)SubKey, 260, (int)v48, 260);
v25 = ReturnCode;
if ( ReturnCode < 0 || (ReturnCode = RegOpenKeyExW(hKey, SubKey, 0, KEY_READ, &hKey)) != 0 )
{
  v34 = ReturnCode;
}
else
{
  cbData = 260;
  v13 = RegQueryValueExW(v26[0], (LPCWSTR)ValueName, 0, &Type, (LPBYTE)Data, &cbData);
```

*Figura 3 – Consulta de registro por dados do CIP.*

Ao explorar a funcionalidade ligada a esse Class ID, foi confirmado que ele é realmente encarregado de consultar o registro e enviar o valor na resposta. Contudo, descobriu-se também que o código que administra essa funcionalidade realiza uma verificação de entrada, permitindo a leitura de valores de registro apenas de chaves específicas da Rockwell.

Foi iniciada a análise da DLL que manipula a classe CIP personalizada para ler e escrever chaves de registro e descobrimos que essa DLL também gerencia duas outras classes CIP personalizadas não documentadas da Rockwell. Decidiu-se investigar essas classes mais profundamente para determinar se elas poderiam ser exploradas para nosso ataque e ajudar a validar nossa hipótese. A primeira classe investigada tinha uma funcionalidade interessante: ela aceita um caminho para um arquivo DLL, um nome de função e um terceiro parâmetro como entrada. Em seguida, ela carrega a DLL usando LoadLibrary e chama a função especificada usando GetProcAddress, passando o terceiro parâmetro como um argumento.

```
mbstowcs_s(&LibFileNameSize, &LibFileName, 260, CipData, 260);
LibHandle = LoadLibraryW(&LibFileName);
if ( !LibHandle )
{
  v36[1] = GetLastError();
  v18 = (_DWORD *)sub_1000E510(v43);
  v29 = (_DWORD *)sub_1000E4B0(v36[1]);
  v19 = (_DWORD *)sub_1000E510(L[redacted]);
  sub_10003C40(-1069539319, 3, *v19, v19[1], *v29, v29[1], *v18, v18[1]);
  goto LABEL_36;
}
mbstowcs_s(&FuncNameSize, FuncName, 260, Src, 260);
ProcAddress = (int (__cdecl *)(char *, BSTR *))GetProcAddressW(LibHandle, FuncName);
if ( ProcAddress )
{
  mbstowcs_s(&FuncArgSize, FuncArg, 260, &CipData[(_DWORD)hLibModule + v36[1] + 2], 260);
  v36[3] = ProcAddress(FuncArg, &FuncNameSize);
```

*Figura 4 – LoadLibrary com base em dados CIP.*

Isso parecia ser um caminho possível para a execução de código arbitrário. Contudo, havia uma ressalva: a classe continha uma função de verificação que checava se o nome da DLL era remotehelper.dll e se o nome da função estava entre os valores predefinidos. Se essas condições não fossem cumpridas, a classe retornaria um erro e não executaria a função. Em seguida, foi realizada a analise da segunda classe encontrada na mesma DLL. Essa classe permitia a leitura e escrita de arquivos no dispositivo.

Ela também continha uma função de verificação, mas era mais flexível: ela apenas verificava se o caminho para leitura/escrita começava com uma string específica. Percebemos que essa classe poderia ser potencialmente explorada para carregar uma DLL maliciosa no dispositivo e colocá-la em praticamente qualquer local.

Com uma compreensão completa das vulnerabilidades, pode ter uma ideia de como um invasor poderia usar as duas classes personalizadas para iniciar o código remotamente no dispositivo. A ideia era compilar uma DLL compatível com o Windows 10 IoT, o sistema operacional do dispositivo. Essa DLL conteria o código que desejávamos executar no dispositivo e seria exportada sob o nome GetVersion, que é um dos nomes de função válidos que podem ser invocados pela classe personalizada. Em seguida, seria utilizada a classe personalizada para carregar nossa DLL no dispositivo, colocando-a em uma pasta aleatória e nomeando-a remotehelper.dll. Finalmente, a executaríamos usando a classe personalizada.

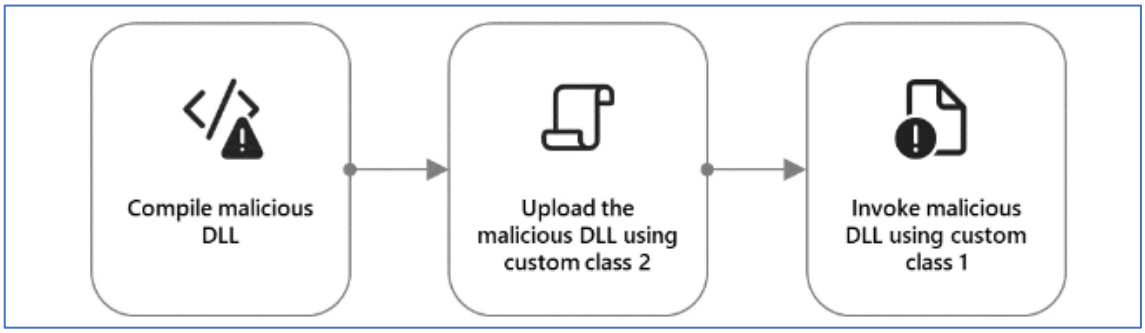



*Figura 5 – Abordagem de exploração.*

Aprofundando-se mais na exploração da vulnerabilidade, decidiu-se utilizar uma função existente no arquivo original remotehelper.dll. Foi descoberto que esse arquivo possuía uma exportação chamada InvokeExe, que permitia a execução de qualquer arquivo executável no dispositivo. No entanto, essa função não estava na lista de nomes de funções válidos para a classe personalizada 1, portanto, não podíamos utilizá-la diretamente. Foi corrigido o arquivo remotehelper.dll e modificamos um dos nomes de exportação válidos para direcionar para a função InvokeExe. Depois, carregamos nossa DLL corrigida no dispositivo, colocando-a em uma pasta diferente da original. Em seguida, utilizamos a classe personalizada 1 para invocar nossa DLL corrigida e executar cmd.exe, o que nos proporcionou um shell de comando no dispositivo

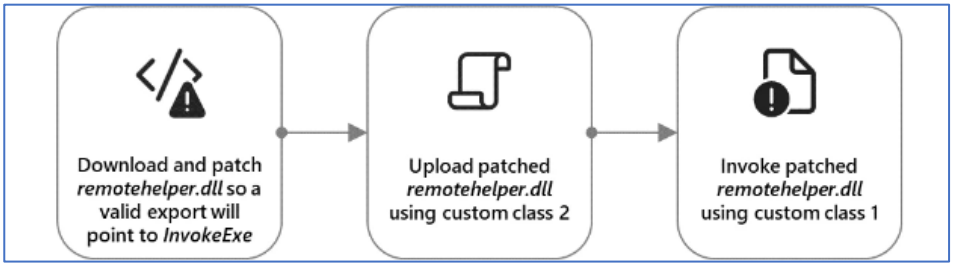



*Figura 6 – Teste de exporação.*

# 3 RECOMENDAÇÕES

A Microsoft recomenda as seguintes medidas para ajudar a proteger as organizações contra ataques que aproveitam as vulnerabilidades do PanelView Plus:

- Aplique patches aos dispositivos afetados na sua rede. O FactoryTalk View ME v12/v13 e o FactoryTalk® Linx v6.20/v6.30 no PanelView Plus são vulneráveis às vulnerabilidades descobertas. É recomendável primeiro identificar os dispositivos na sua rede que são impactados por essas vulnerabilidades. Também é recomendável instalar os seguintes patches no dispositivo:
    - PN1645 | FactoryTalk View Machine Edition vulnerável à execução remota de código
    - PN1652 | FactoryTalk® Linx vulnerável a negação de serviço e divulgação de informações
- Certifique-se de que todos os dispositivos críticos, como PLCs, roteadores, PCs, etc., estejam desconectados da Internet e segmentados, independentemente de executarem ou não o FactoryTalk View da Rockwell.
- Limite o acesso aos dispositivos CIP somente aos componentes autorizados.

Para ajudar a identificar dispositivos afetados, a Microsoft lançou uma ferramenta para escanear e realizar investigação forense em dispositivos Rockwell Rslogix como parte de seu arsenal de ferramentas de código aberto disponíveis no GitHub.

# 4 REFERÊNCIAS


- **Heimdall *by* ISH Tecnologia**
- Microsoft
- Thehackernews
- NVD

# 5 AUTORES

- **Leonardo Oliveira Silva**

heimdall
security research
A DIVISION OF ISH