Protegendo Pontos Fracos em Arquiteturas Serverless

Riscos e Recomendações

Alfredo de Oliveira

Protegendo Pontos Fracos em Arquiteturas Serverless: Riscos e Recomendações

Publicado pela
Trend Micro Research

Escrito por
Alfredo de Oliveira
Pesquisador Sênior de Ameaças

Imagem de estoque usada sob licença de Shutterstock.com

Conteúdo

A nuvem pública capacitou as empresas a alcançar novos patamares digitais, permitindo-lhes criar operações dinâmicas e escaláveis. Para suas diversas necessidades dinâmicas e flexíveis, existem diferentes opções de computação disponíveis para as empresas escolherem[1]. Uma delas é o modelo serverless.

A computação serverless é um tipo de modelo de execução de computação em nuvem que permite às empresas usar o poder computacional de um provedor de serviços de nuvem (CSP), como Amazon Web Services (AWS). Ela permite que empresas tirem proveito de uma redução adicional nas despesas gerais relativas às operações e manutenção do servidor e aos processos associados, como gerenciamento de patches, escalonamento, e disponibilidade. Com a computação serverless, as empresas podem se concentrar na criação de aplicações e produtos essenciais, ao invés de usar recursos humanos para manter e proteger a infraestrutura do servidor. Isso significa que as empresas que optam por trabalhar serverless se beneficiam de maior flexibilidade, automação, economia e agilidade.

De capacitar e dimensionar sites e aplicações em uma questão de minutos[2] sem exigir que os adotantes se preocupem com a infraestrutura, permitindo que as organizações iterem o software mais rapidamente usando a metodologia de integração contínua e deploy contínuo (CI/CD), a tecnologia serverless está permitindo que as organizações tenham a velocidade e a eficiência de que precisam para impulsionar a inovação e melhorar o negócio.

O modelo serverless é considerado relativamente mais seguro do que outros modelos de nuvem porque, por exemplo, no caso do AWS Lambda, a AWS cuida da infraestrutura subjacente, do sistema operacional e da plataforma da aplicação. Isso, porém, não significa que proteger o modelo serverless é responsabilidade exclusiva da AWS. Os usuários do AWS Lambda são responsáveis por proteger seu código, o armazenamento e a acessibilidade de dados confidenciais e o gerenciamento de identidade e acesso (IAM) em relação ao serviço AWS Lambda e em suas funções. Resumindo, os serviços que os usuários optam por usar determinam suas responsabilidades. O modelo serverless também exige que os clientes entendam sua responsabilidade em manter IAM adequado, armazenamento e acessibilidade de dados críticos e qualidade de código. Os profissionais de CloudOps e DevOps precisam ser responsáveis pela configuração adequada de elementos, como IAM e armazenamento de dados críticos, à medida que configuram os serviços de nuvem, além de garantir que estejam implantando código seguro.

Este artigo de pesquisa visa esclarecer as considerações de segurança em ambientes serverless e fornecer recomendações que podem ajudar a garantir que as implantações serverless sejam mantidas o mais seguras possível.

Arquiteturas Serverless

A computação serverless refere-se à tecnologia que oferece suporte a serviços de backend e permite que as empresas aproveitem a transferência de certas responsabilidades para CSPs, como a AWS, incluindo gerenciamento de capacidade, patching e disponibilidade. Com a computação serverless, as empresas podem criar aplicações backend sem estar diretamente envolvidas na disponibilidade e escalabilidade. Além da acessibilidade relativa desse tipo de computação, sua arquitetura também permite que as empresas escrevam e implantem código sem se preocupar com o gerenciamento e a segurança da infraestrutura subjacente — daí o termo "serverless", sem servidor.

Como os componentes de infraestrutura de computação da tecnologia serverless — incluindo o gerenciamento do hardware, o sistema operacional e as partes do software — são gerenciados pelos próprios CSPs, esses componentes serverless também são protegidos por sua segurança. Embora os CSPs tenham uma fatia maior da torta de responsabilidade pela segurança no que diz respeito a essa computação, o modelo de responsabilidade compartilhada[3] também se aplica a esses tipos de serviços. Por exemplo, no caso de um serviço categorizado como infrastructure as a service (IaaS), o usuário deve implementar todas as medidas de segurança para proteger o sistema operacional e o software da aplicação. Ao expor qualquer serviço à Internet, o usuário também deve gerenciar suas próprias regras de política de firewall, bem como ele também é responsável por sua aplicação e dados.

Em uma arquitetura serverless, que também é referida em termos de serviços abstratos, os principais CSPs como a AWS garantem uma infraestrutura segura para seus usuários a fim de que eles possam se concentrar no gerenciamento e proteger seus próprios dados — incluindo código de aplicação e dados binários, bem como classificação de ativos e fornecimento de permissões.

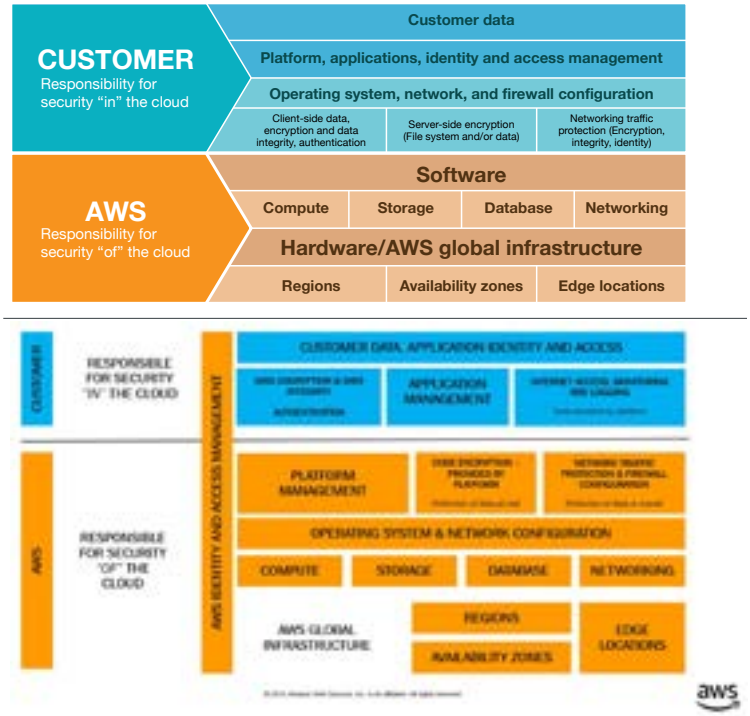

Figura 1. O modelo clássico de responsabilidade compartilhada (parte superior) e o modelo de responsabilidade compartilhada para o AWS Lambda (parte inferior)

Créditos da imagem: AWS[4, 5]

Os CSPs têm dois princípios básicos de design: privilégio mínimo6 e negação padrão. O privilégio mínimo fornece uma camada de segurança, já que a maioria das aplicações serverless é construída em um conjunto heterogêneo de serviços, enquanto a negação padrão permite a interação proposital dos microsserviços nos quais as aplicações são construídas. Por sua vez, no modelo de responsabilidade compartilhada, os administradores e usuários devem seguir esses princípios ao projetar e aplicar políticas e permissões para os serviços usados em suas aplicações serverless.

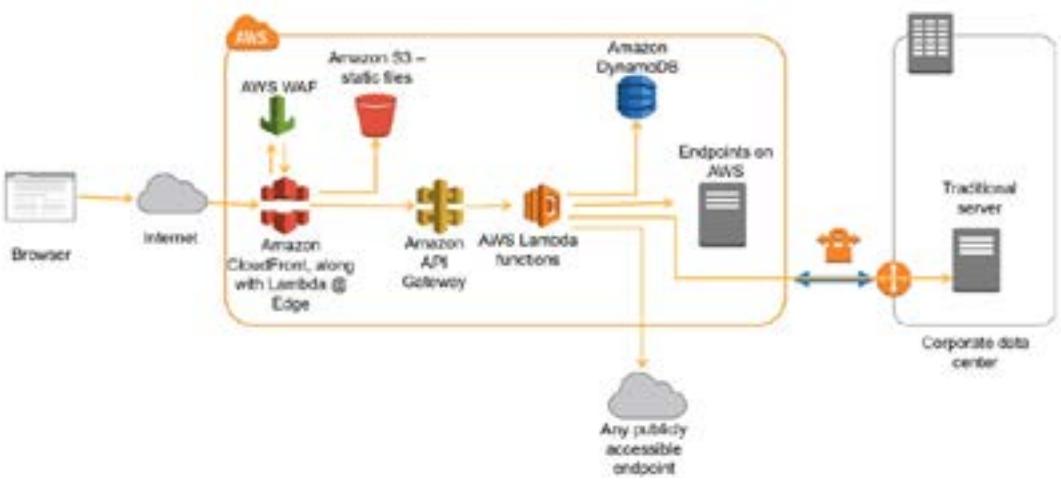

Figura 2. Um exemplo de pipeline de aplicação web serverless

Créditos da imagem: AWS Compute Blog[7]

Serviços Conectados em uma Arquitetura Serverless

Entender como uma arquitetura serverless opera envolve a compreensão dos diferentes serviços envolvidos nela. O escopo deste documento inclui serviços oferecidos pela AWS, que é o principal fornecedor de soluções serverless.[8](De acordo com uma pesquisa recente, mais de 80% dos desenvolvedores estão cientes das soluções AWS Lambda, enquanto 51% usam essas soluções em suas implantações serverless.[9])

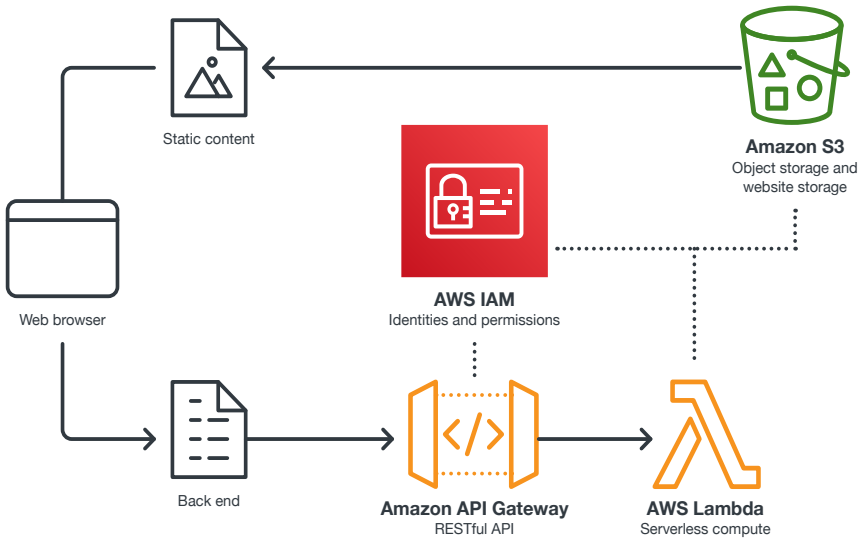

Figura 3. Um exemplo de serviços interconectados em uma arquitetura serverless

Amazon S3

O Amazon Simple Storage Service (Amazon S3)10 é um serviço de armazenamento de objetos para uma quantidade escalonável de dados que oferece suporte a uma variedade de casos de uso, como aplicações móveis, análise de big data e dispositivos de Internet das Coisas (IoT). O Amazon S3 permite que as empresas gerenciem objetos, que são armazenados em buckets por meio de APIs.

Ao criar buckets do Amazon S3, um usuário tem a opção de aplicar privilégio mínimo para seus objetos e buckets. Um administrador com permissões do Amazon S3 define as políticas para os usuários, e estes podem receber permissões suficientes para gerenciar permissões adicionais ou permissões podem ser bloqueadas para que um bucket possa ser usado apenas de determinadas formas. Um usuário com permissões pode gerenciar um intervalo específico naquela região por meio da console web ou da API remota. Para ambos os cenários, o usuário deve ser autenticado para ter acesso. Contudo, se necessário ou desejado, o usuário também tem a opção de disponibilizar um objeto ou bucket, dependendo da política definida pelo administrador. Pode ser disponibilizado ao público ou a outros serviços ou utilizadores internos.

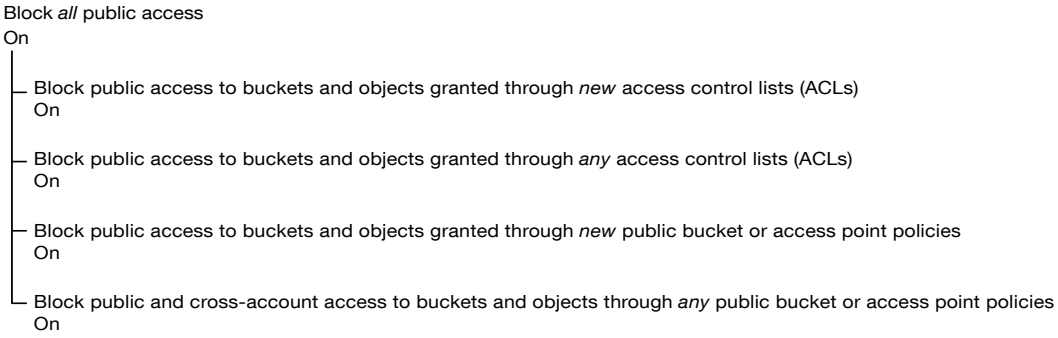

Figura 4. A configuração padrão de acesso público do bucket do Amazon S3

AWS Lambda

Um dos serviços serverless mais populares hoje[11], o AWS Lambda permite que as empresas executem código sem o incômodo de provisionar e manter um servidor. Com ele, os desenvolvedores podem executar o código e pagar apenas pelo número de instâncias quando o código é acionado. Com um conjunto selecionado de linguagens de programação, o AWS Lambda permite que eles se concentrem em suas tarefas sem ter que gerenciar hardware ou garantir que o sistema operacional e todas as aplicações instaladas estejam atualizados. Se uma empresa usa o AWS Lambda apenas, ela pode se concentrar apenas na escrita de código seguro. Se o AWS Lambda for usado junto com outro serviço, uma empresa deve estar ciente das permissões concedidas. O usuário que definiu as permissões deve aplicar a abordagem de privilégio mínimo para este serviço, o que significa que se o AWS Lambda vai funcionar com outros serviços, as permissões devem ser concedidas manualmente.

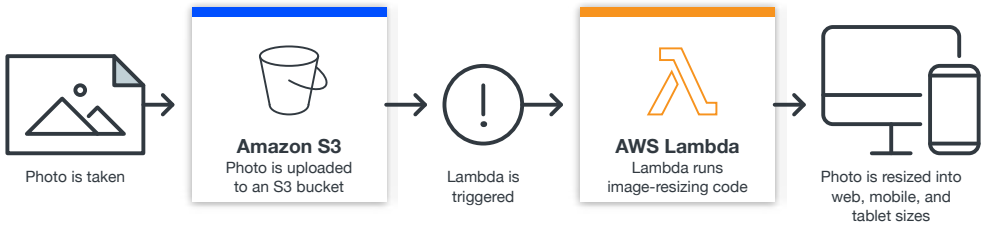

Figura 5. Um caso de uso para AWS Lambda

Amazon API Gateway

O Amazon API Gateway é um serviço que permite a criação, publicação, manutenção, monitoramento e proteção de API de maneira fácil e eficiente.[12] Como o próprio nome indica, ele atua como um portal para aplicações, permitindo que elas acessem funcionalidades de serviço de backend ou dados usando APIs RESTful e APIs WebSocket. Assim como no modelo de pagamento do AWS Lambda, o Amazon API Gateway permite que as empresas paguem apenas pelas chamadas de API recebidas e pela quantidade de dados transferidos. Ele também pode atuar como uma ponte que conecta uma implantação a outros serviços da Amazon.

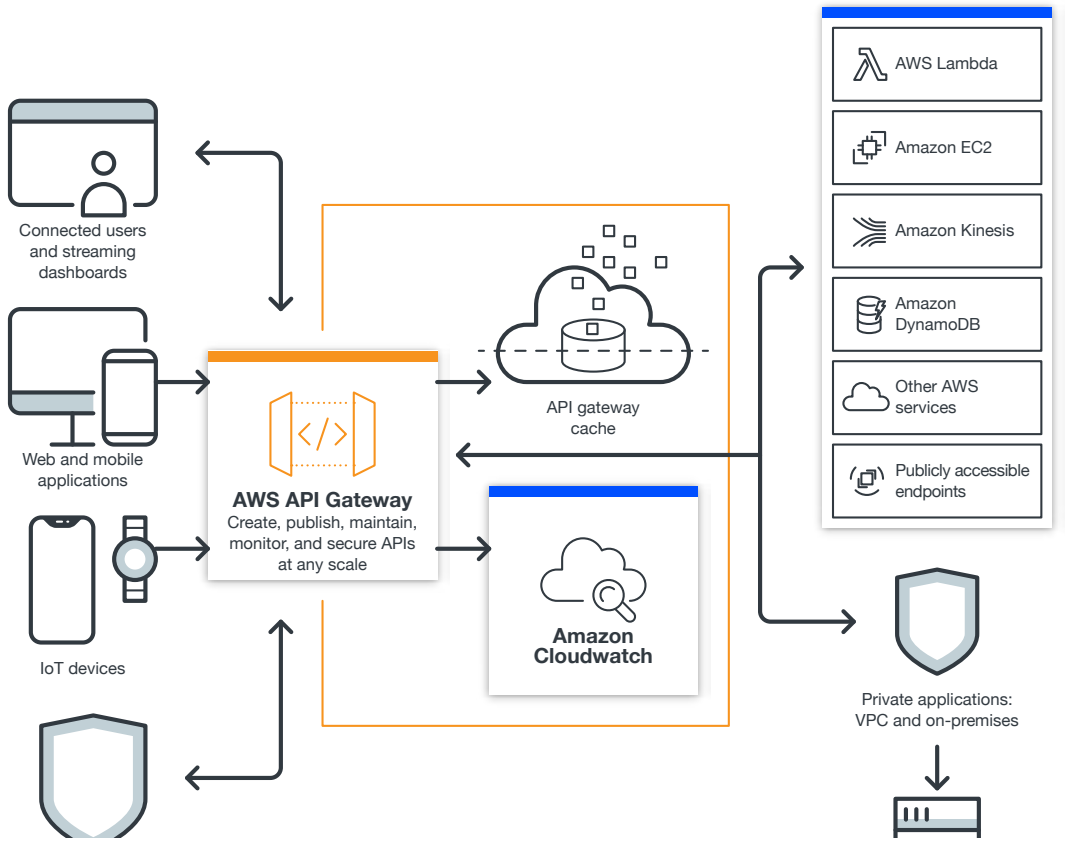

Figura 6. Um caso de uso para Amazon API Gateway

AWS IAM

O AWS Identity and Access Management (AWS IAM) permite que os desenvolvedores criem credenciais e permissões de segurança e as atribuam aos usuários. Usando o AWS IAM, uma empresa também pode permitir a federação de identidade para habilitar identidades existentes dentro da organização a fim de obter acesso ao AWS Management Console, chamar APIs e acessar recursos, mesmo sem a criação de usuários AWS IAM exclusivos.

O usuário que cria as políticas do AWS IAM deve definir e aplicar a abordagem de privilégios mínimos para garantir que os serviços não sejam acessíveis, a menos que o acesso seja explicitamente concedido pelo usuário. As funções do IAM também podem ser usadas como uma prática recomendada. Uma função do IAM não está vinculada exclusivamente a uma única pessoa e credenciais de longo prazo. Ao invés disso, ela fornece credenciais temporárias para uma sessão de funções.[13]

Configurações incorretas e práticas de codificação inseguras

Os usuários dos serviços mencionados acima devem definir políticas para usar a abordagem de privilégios mínimos como uma prática recomendada e devem atribuir e verificar os privilégios diligentemente para uma melhor postura de segurança. No entanto, uma combinação complexa de serviços pode ser difícil para os usuários resolverem manualmente. Nesta seção, discutimos e demonstramos configurações incorretas e riscos que os usuários podem introduzir ou ignorar ao proteger serviços específicos serverless da AWS. Para ajudar a evitar essas configurações incorretas e riscos, a AWS fornece a seus usuários o AWS Well-Architected Framework[14], um conjunto de práticas recomendadas de arquitetura para projetar e executar sistemas em nuvem seguros e eficientes.

Amazon S3

A Amazon oferece um guia abrangente para manter os buckets do Amazon S3 seguros[15], incluindo recomendações sobre a ativação da exclusão de autenticação multifatorial para evitar exclusões acidentais de buckets e objetos individuais dentro de buckets. No entanto, mesmo com essa orientação, os usuários ainda podem encontrar armadilhas de segurança, como as seguintes.

Deixando um Bucket Aberto e/ou Acessível Publicamente

Agentes mal-intencionados podem abusar de um bucket aberto ou acessível publicamente para procurar dados confidenciais. Houve casos em que dados críticos e confidenciais foram deixados abertos devido a configurações incorretas, como quando um banco de dados contendo mais de 500.000 documentos jurídicos e financeiros confidenciais e privados foi exposto[16]. Em fevereiro de 2020, um repositório exposto associado a uma aplicação baseada em nuvem chamada JailCore foi descoberto vazando mais de 36.000 registros de presidiários

de várias instalações correcionais nos EUA.[17] Desde então, isso foi abordado pelo JailCore e o repositório foi devidamente protegido.

Infelizmente, muitos buckets do Amazon S3 são deixados sem segurança e abertos ao público, apesar das práticas recomendadas. Durante nossa pesquisa, pudemos encontrar alguns buckets abertos com dados confidenciais e outros que não estavam completamente abertos, mas que estavam indexando dados acessíveis.

Figura 7. Um bucket de e-commerce encontrado por meio de uma pesquisa Shodan que mostra detalhes do pedido, incluindo informações do cliente

Criação de Código Ruim ou Exposto

Os buckets do Amazon S3 são usados não apenas como armazenamento em nuvem, mas também como servidores web estáticos simples que podem interagir com outros serviços, como AWS Lambda, Amazon API Gateway, Amazon CloudFront e DynamoDB

Figura 8. Um bucket do Amazon S3 hospedando um site estático

A AWS sempre foi clara sobre o que os buckets do Amazon S3 devem hospedar: conteúdo estático. O Amazon S3 não deve ser usado para hospedar conteúdo dinâmico.[18] Portanto, os scripts do lado do servidor não devem ser usados. No entanto, muitos usuários não estão seguindo o uso recomendado para buckets do Amazon S3 e estão se abrindo para riscos maiores. Não é difícil encontrar sites hospedados em buckets do Amazon S3 que implementam linguagens de script do lado do servidor, como PHP, JSP ou ASP.NET. Como não é interpretado pelo servidor web, o código será exposto quando o código-fonte do site for acessado.

Durante nossa pesquisa, encontramos uma página da web, hospedada em um bucket do Amazon S3, que contém o código PHP que não está sendo interpretado por não ser estático. Como resultado, quando o código-fonte desta página é consultado, qualquer pessoa pode ver que a página usa ferramentas de linha de comando como curl ou wget. Isso pode expor dados confidenciais, como credenciais ou partes de código que não devem ser vistas pelo público, como caminhos de recursos e lógica de programação. Isso também pode tornar mais fácil para agentes mal-intencionados encontrar e explorar vulnerabilidades no código, usando técnicas como script entre sites (XSS) e injeção de SQL (SQLi).[19], [20]

```
<?php
header('HTTP/1.1 503 Service Temporarily Unavailable');
header('Status: 503 Service Temporarily Unavailable');
header('Retry-After: 3600');
?>
<!DOCTYPE html>
<html>
<head>
<link href='https://fonts.googleapis.com/css?family=Roboto' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Roboto:300' rel='stylesheet' type='text/css'>
<link rel="icon" href="favicon.ico" sizes="32x32" />

    <title>Ecommerce Software
    <style type="text/css">
body {
    margin: 0;
    padding: 0; background: #fbfbfb;
    font-family: 'Roboto', sans-serif;
}
.container {
    width: 75%;
    margin: 0 auto;
}
```

Figura 9. Uma página da web, hospedada em um bucket do Amazon S3, que encontramos à solta, mostrando o código PHP expostoexposed PHP code

AWS Lambda

Além das recomendações de segurança da AWS que tratam de proteção de dados, configuração e compliance[21], usuários também podem se beneficiar de certas funções do AWS Lambda que visam proteger os dados dos usuários, bem como garantir que os custos não saiam do controle. No entanto, os usuários ainda podem encontrar incidentes de segurança devido a práticas não seguras, como as seguintes.

Criação de Código Ruim ou Vulnerável

Como as funções do AWS Lambda devem ser acionadas para executar código e fornecer um resultado, há vários cenários em que a codificação incorreta de uma função do AWS Lambda pode permitir que um agente malicioso use técnicas comuns de injeção de código, como quando uma função lida com dados de entrada do usuário. Além disso, os dados de entrada podem não vir de uma navegação normal ou de uma chamada de API HTTP. As funções do AWS Lambda podem ser implantadas em vários cenários diferentes e injeções maliciosas podem ser enviadas de diferentes gatilhos, como aqueles do Amazon Simple Notification Service (SNS)[22], de e-mail e até mesmo de IoT. Nesses casos, um firewall de aplicação da web (WAF) regular no nível da API não é capaz de proteger as funções do AWS Lambda contra injeções maliciosas.[23]

Deixando Dados Sensíveis Expostos

Ao executar funções do AWS Lambda, existem algumas variáveis de ambiente que são criadas para passar configurações específicas do ambiente, como autorizações, tamanho dos recursos e o host do contêiner que está executando o código. Para autorizar ou gerenciar outros serviços ou camadas, os usuários podem criar suas próprias variáveis com usuários, senhas, tokens de autorização e parâmetros de aplicação. Como o contêiner irá expirar após a execução, os dados não são mantidos.

No entanto, se o código de função do AWS Lambda estiver configurado para retornar variáveis e estiver acessível a partir de serviços externos, ou se um agente malicioso conseguir injetar comandos na função, dados podem vazar.

Figura 10. Variáveis mostradas de um resultado de função AWS Lambda

Salvar Credenciais como Variáveis

Ao verificar onde as funções do AWS Lambda são executadas dentro de uma máquina virtual, é surpreendente que as credenciais e os segredos — os tokens e as chaves usadas pelas funções para autenticação — estejam sendo armazenados como variáveis. Em uma atividade mal-intencionada hipotética, se o agente conseguir comprometer um contêiner e baixar, executar e rodar a ferramenta AWS Command Line Interface (AWS CLI),[24] ele poderá obter acesso à conta de um usuário sem solicitar quaisquer credenciais adicionais dado que o AWS CLI ferramenta facilita um login único.

Usando Bibliotecas Vulneráveis

Ao desenvolver uma função com uma linguagem de programação preferida, o usuário deve ter em mente que as bibliotecas importadas, bem como o próprio código, devem ser seguros. Usar componentes com vulnerabilidades conhecidas é um dos 10 principais riscos das aplicações web, de acordo com o Open Web Application Security Project (OWASP).[25]

Copiando Maus Exemplos de Código a Partir de Repositórios Online

Desenvolver uma função AWS Lambda pode ser diferente de desenvolver uma aplicação que seria executada dentro de um ambiente do qual o usuário tem controle total. A AWS fornece documentação essencial e códigos básicos para ajudar os usuários a se familiarizarem com o serviço.[26] No entanto, os usuários que procuram códigos mais complexos ou elaborados em plataformas de compartilhamento de código online devem ter cuidado antes de promover tais códigos para produção, pois eles podem ter sido desenvolvidos usando práticas inadequadas ou podem conter componentes vulneráveis. A pesquisa apontou como o código compartilhado em sites de perguntas e respostas de programação online pode ser de baixa qualidade e abrigar vulnerabilidades.[27]

Figura 11. Amostra de código do GitHub

Persistência de Arquivo

Os arquivos podem ser gravados na pasta /tmp dentro de um ambiente de execução do AWS Lambda. Os arquivos gravados ali podem ter permissão de execução, o que significa que se um agente malicioso explorar com sucesso um código inválido, ele poderá salvar suas ferramentas e scripts nesta pasta e executá-los a partir daí. Com base em nossos testes, a menos que o código seja alterado na console do AWS Lambda, os arquivos gravados lá podem durar até 12 minutos. Um “problema” colateral é que, se um contêiner tiver acesso à internet (por meio de uma conexão de internet rápida, apenas para fins de métrica), uma função configurada com um tempo limite de 45 segundos pode baixar um arquivo de 334 megabytes. Se um invasor estiver usando a função como um ponto de articulação, isso permitirá que ele baixe mais facilmente ferramentas de hacking para uso posterior.

Figura 12. Um exemplo de um grande arquivo escrito na pasta /tmp dentro de uma função AWS Lambda que é persistente por 12 minutos

Amazon API Gateway

A AWS criou recursos de segurança para que os usuários do Amazon API Gateway tirem proveito ao desenvolver e implementar suas políticas de segurança. A AWS também fornece considerações úteis para diferentes tipos de ambientes corporativos.[28] Apesar disso, os usuários ainda podem expor endpoints do Amazon API Gateway de forma insegura. Uma possível causa para isso é a confiança dos usuários na segurança padrão.

O Amazon API Gateway é um serviço que adiciona outra camada que ajuda a filtrar solicitações e evitar a exposição direta de serviços serverless. Embora execute a filtragem de entrada inicial, sua configuração padrão pode se beneficiar de recursos de segurança aprimorados que podem e devem ser implementados desde o início, como a autenticação. Deve-se observar, entretanto, que embora o Amazon API Gateway forneça proteção adicional, ele não deve ser visto de forma alguma como um substituto para um WAF.

Um endpoint aberto e exposto do Amazon API Gateway pode ser abusado como o ponto de entrada em um ataque que visa comprometer o serviço por trás dele, ser acionado para causar um ataque de negação de serviço (DoS)[29] ou até mesmo aumentar a fatura de uma empresa se uma função AWS Lambda for consultada com frequência ou incessantemente.

AWS IAM

A AWS fornece diretrizes sobre como ajudar a proteger melhor os recursos da AWS que

podem ser acessados por meio do AWS IAM. No entanto, os usuários ainda podem se deparar com falhas de segurança devido às seguintes práticas não seguras.

Modificar Um Serviço Usando Credenciais Salvas Como Variáveis e a Ferramenta AWS CLI

Conforme observado anteriormente, as credenciais são armazenadas dentro da infraestrutura AWS Lambda como variáveis. Se um agente malicioso tiver sucesso em explorar uma função e carregar a ferramenta AWS CLI, a ferramenta pode usar as credenciais salvas para gerenciar serviços AWS, incluindo o próprio AWS Lambda.

Usando Configurações Inadequadas ou Muito Permissivas

Muitas vezes encontramos projetos em plataformas de compartilhamento de código on-line em que os desenvolvedores usam políticas que estão na extremidade oposta do espectro da política de privilégios mínimos — políticas que são muito permissivas, garantindo assim que todo o sistema se comunique e todos os serviços sejam responsivos um para o outro. Como o AWS IAM confirma o acesso aos serviços e recursos da AWS e, portanto, determina com quais serviços o AWS Lambda pode se comunicar, este exemplo também se enquadra nas considerações de segurança do AWS Lambda — a cópia de exemplos de código inválido de repositórios online representa uma falha de segurança.

Figura 13. Um exemplo de código sendo explorado ativamente com uma política bem escrita implementando privilégios mínimos (esquerda) e outro exemplo com uma política muito permissiva (direita)

Dependendo do tamanho de um projeto, lidar com todas as políticas e permissões manualmente é quase humanamente impossível, pois pode se tornar muito complicado e confuso. Por esse motivo, as empresas com workloads e aplicações complexas da

AWS devem ter uma ferramenta de auditoria centralizada para IAM e para gerenciar chaves de acesso, credenciais de segurança e níveis de permissão.[30] Funcionalidades do AWS IAM, como consultor de acesso[31] e o Access Analyzer[32] também podem ajudar no monitoramento de políticas e permissões.

Possíveis Cenários de Comprometimento e Ataque

A AWS fornece mecanismos de segurança usados em serviços serverless para que os usuários instalem e configurem. No entanto, os agentes mal-intencionados procuram várias maneiras de aproveitar os erros comuns do usuário, as configurações incorretas e até mesmo um dos pontos fortes do modelo serverless — sua natureza distribuída — para prosseguir com suas atividades.

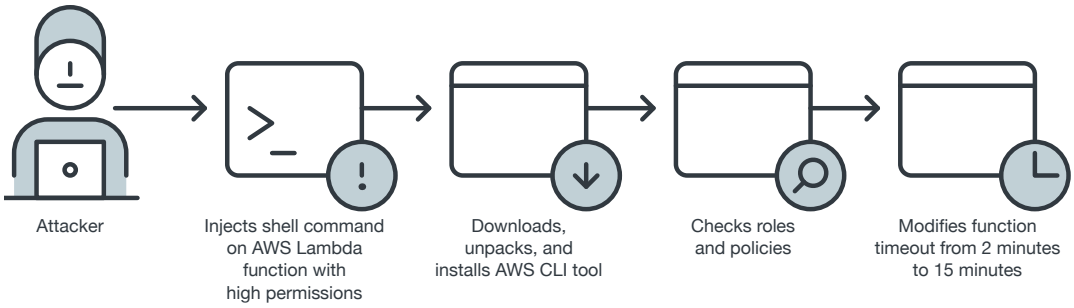

Figura 14. Uma cadeia de ataque envolvendo uma função AWS Lambda com altas permissões

Roubo de Credencial e Conta

As funções, ao acessar recursos seguros, precisam de segredos. Quando um agente malicioso consegue obter uma aplicação comprometida em um sistema serverless, ele pode obter e usar segredos contendo credenciais seguras para conseguir acesso a recursos críticos ou assumir o controle de toda a conta — especialmente porque os segredos são convertidos em texto simples quando não estão em uso.

Dados Confidenciais e Roubo de Código

Quando os buckets do Amazon S3 são feitos para hospedar conteúdo dinâmico para uma aplicação da web, pode ser fácil para os agentes mal-intencionados ver dados confidenciais

e partes críticas do código simplesmente consultando o site. Isso pode permitir que eles usem dados confidenciais, como nomes de usuário e senhas, e ferramentas de linha de comando para realizar o reconhecimento ou procurar vulnerabilidades que possam explorar. Quando os dados confidenciais do usuário são expostos publicamente, isso também pode causar danos financeiros e de reputação às empresas.

Escalonamento de Privilégios

Quando os usuários não definem e configuram adequadamente as permissões anexadas a um serviço, pode acontecer o escalonamento de privilégios. Isso pode permitir que um usuário comprometido com poucos privilégios altere a senha de um usuário com altos privilégios. O mesmo vale para políticas de permissão de função configuradas incorretamente, o que pode permitir que um usuário mal-intencionado crie uma nova versão de política que pode, por sua vez, permitir a alteração de permissões em uma política. Se as permissões de função não estiverem definidas com segurança, o usuário malicioso pode receber privilégios totais de administrador. É importante observar que, embora demore para criar uma lista abrangente de funções que indique quais usuários e serviços na arquitetura podem passar, essa lista ajuda a garantir um sistema mais seguro e sem configurações incorretas.

Uso Indevido de Recursos que Geram Custos para o Proprietário da Conta

Como os serviços serverless, tais como o AWS Lambda, podem fornecer elasticidade de recursos e escalabilidade automatizada, agentes mal-intencionados que buscam causar custos financeiros para as empresas por qualquer motivo podem introduzir código que, por exemplo, consultariam incessantemente as funções do AWS Lambda, aumentando assim os custos. Esses agentes podem basicamente alterar tudo o que é gerenciável com a ferramenta AWS CLI, incluindo a alocação de memória de uma função AWS Lambda, o que pode aumentar ainda mais os gastos.

Persistência

As funções serverless têm informações de configuração associadas (nome, descrição, ponto de entrada e requisitos de recursos) e são efêmeras, ou seja, têm apenas alguns segundos ou minutos de vida. Eles também são projetados para serem stateless, permitindo o lançamento rápido de quantas cópias de uma função forem necessárias para dimensionar a taxa de eventos de entrada. Com cada chamada de função, uma nova instância dela é criada. A primeira vez que uma função é chamada, conhecida como “inicialização a frio”, espera-se que tenha um pouco de latência. Para evitar latência, as funções precisam permanecer em ocultas — o que também conhecido como “warm starts” — em que a mesma caixa

de proteção de função ou contêiner para diferentes invocações são reutilizados. As warm starts podem ser aproveitadas por agentes mal-intencionados que já comprometeram uma instância do AWS Lambda para enviar solicitações periodicamente a fim de evitar que a instância seja desativada.

Outras Considerações de Segurança em Deploys Serverless

A Amazon fornece recursos de segurança para seus serviços e orientações para os usuários considerarem a fim de proteger seus respectivos ambientes. Porém, é seguro dizer que sempre haverá espaço para melhorar a segurança de serviços serverless. Nesta seção, discutimos oportunidades de aprimorar a segurança dos serviços conectados de uma arquitetura serverless. (Sugerimos melhores práticas e outras medidas de segurança que tratam desses problemas na próxima seção).

Amazon S3

Para manter os buckets do Amazon S3 seguros, novos S3 buckets são gerados com acesso público bloqueado por padrão.[33] A documentação da AWS também aponta fortes recomendações de segurança que os usuários podem aplicar para garantir que seus buckets do Amazon S3 sejam protegidos usando configurações da AWS.[34] Isso inclui garantir que os buckets sejam privados, implementar protocolos de autenticação e adicionar mais camadas de proteção aos buckets para restringir ainda mais quem pode acessá-los de cada ponto de entrada.

Com relação à solução de problemas de abuso de bucket do Amazon S3, algo que torna o processo mais desafiador é a falta de um recurso para fácil registro. A opção de logs padrão que está disponível oferece suporte ao consumo e uso de armazenamento básico, mas não permite que os usuários vejam quem está acessando seus buckets do Amazon S3. O que os usuários podem fazer é ativar o registro em nível de objeto, que armazena operações de API em nível de objeto do Amazon S3, como GetObject, DeleteObject e PutObject no AWS CloudTrail.[35]

Além disso, se um usuário estiver hospedando um site em um bucket do Amazon S3, há uma opção para registrar o acesso do usuário e enviar todos os logs para uma pasta dedicada

dentro de um bucket do Amazon S3. Essa é uma boa opção que pode resolver a dificuldade de solução de problemas, mas também pode criar um grande número de arquivos que contêm logs de processo fragmentados. Essa é uma boa opção que pode resolver a dificuldade de solução de problemas, mas também pode criar um grande número de arquivos que contêm logs de processo fragmentados. Para melhor visibilidade, os usuários podem usar um serviço de consulta interativa serverless para analisar dados (como o Amazon Athena[36]) e um serviço de gerenciamento de postura de segurança em nuvem com extensos recursos de relatórios.

Os buckets do Amazon S3 também deixam pegadas ou assinaturas em cabeçalhos HTTP quando usados, facilitando a identificação de que um usuário está interagindo com o Amazon S3. Isso também torna possível a consulta de buckets abertos ou expostos do Amazon S3 no ambiente selvagem. Em abril de 2020, relatamos como agentes mal-intencionados tiraram proveito de buckets do Amazon S3 graváveis para todo o mundo; com base em nossa telemetria, a maioria das atividades em 2019 envolveu injeção de código malicioso e exfiltração de dados.[37]

Figura 15. Um cabeçalho HTTP mostrando uma solicitação para uma página hospedada em um bucket do Amazon S3

AWS Lambda

O processamento de dados de entrada não filtrados de várias fontes pode ser considerado um dos pontos de melhoria de segurança ao usar o AWS Lambda. É importante observar, porém, que os usuários deste serviço, e não os CSPs, são responsáveis por garantir que o código que eles usam esteja em compliance com as práticas recomendadas de codificação. Proteger as funções do AWS Lambda pode ser desafiador quando os dados de entrada não são bem higienizados ou usam bibliotecas vulneráveis, precisamente porque os dados de entrada podem vir de várias fontes.

Amazon API Gateway

Ao criar um endpoint no Amazon API Gateway, a configuração padrão não inclui o uso de qualquer tipo de autenticação ou chaves de segurança, o que pode levar ao uso malicioso ou abuso de APIs.

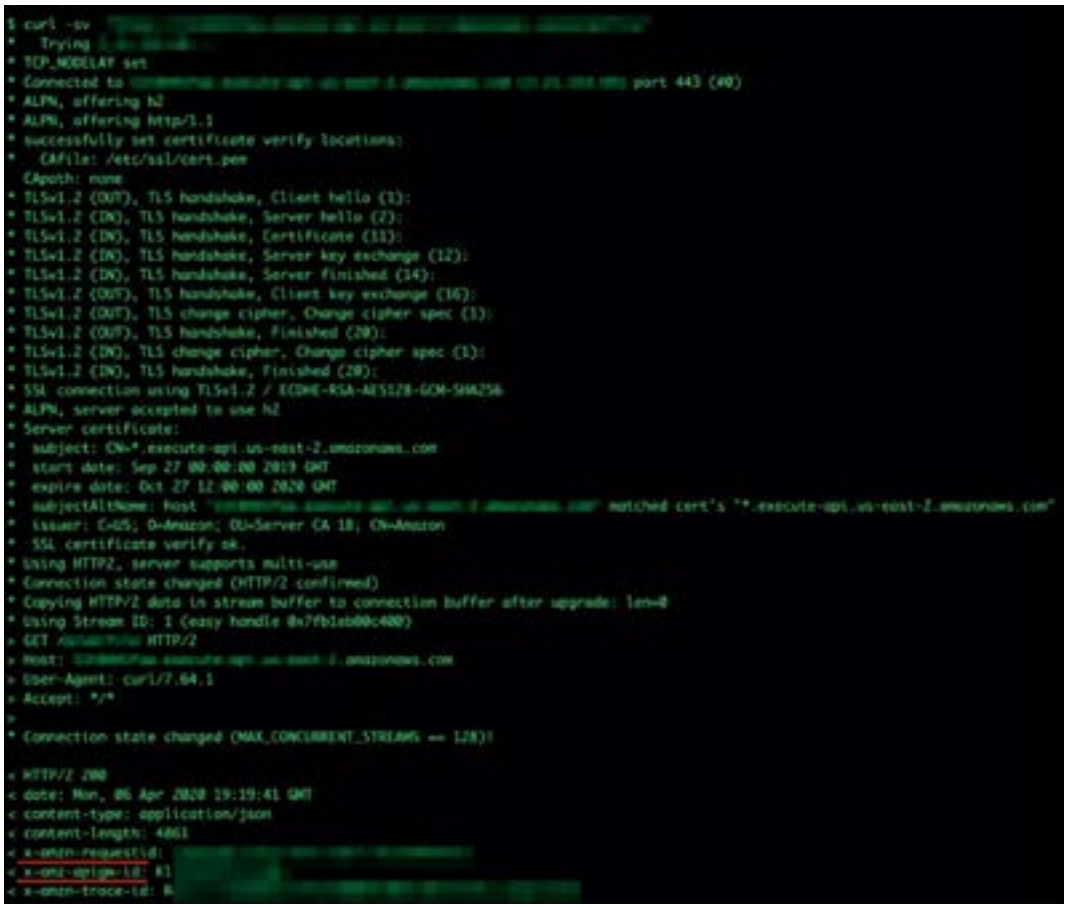

Figura 16. Um cabeçalho HTTP mostrando as informações x-amzn-RequestId e x-amz-apigw-id de um bucket do Amazon S3

AWS IAM

Conforme o ambiente de um usuário cresce, as políticas e as funções também aumentam em complexidade. Rastrear e gerenciar isso pode ser um desafio sem o auxílio de uma ferramenta de auditoria externa.

Medidas de Segurança para Serviços Serverless

Os serviços serverless se tornaram ferramentas de negócios essenciais e mantê-los seguros deve ser a prioridade máxima não apenas para as empresas que os utilizam, mas também para as pessoas e organizações que dependem das aplicações dessas empresas. A seguir estão algumas práticas recomendadas e soluções de segurança que podem ajudar a manter os serviços serverless seguros.

Amazon S3

É seguro dizer que a AWS faz sua parte no modelo de responsabilidade compartilhada. Ela também ajuda os usuários a cumprir sua parte do modelo, tornando mais difícil para os administradores expor seus buckets sem saber o que estão fazendo. Assim, os administradores precisam verificar constantemente se as configurações estão definidas corretamente, usando serviços como AWS Config e AWS CloudTrail, ou consultando as APIs do Amazon S3. Eles também podem usar ferramentas de terceiros que verificam o conteúdo dos buckets do Amazon S3 e suas políticas para verificar o acesso e as configurações de segurança e compliance[38] — com DNS (Domain Name System) de nomes de bucket do Amazon S3, acesso público de leitura/gravação e criptografia de dados via TLS (Transport Layer Security) durante o transporte.

AWS Lambda

A AWS garante que os ambientes de execução usados nas funções do AWS Lambda sejam efêmeros. Isso significa que os ambientes de execução estão constantemente sendo substituídos por novos. Por causa disso, os dados salvos em um contêiner permanecerão lá apenas por um curto período, tornando mais difícil para os agentes mal-intencionados realizarem escalonamento de privilégios ou movimento lateral.

Outro recurso digno de nota do AWS Lambda é a função timeout.[39] Ele visa proteger os usuários de atividades maliciosas que causam um faturamento involuntariamente alto. Cada

linguagem de programação tem seu próprio tempo limite de função padrão, que normalmente é definido como baixo. Por exemplo, ao criar uma aplicação usando Python, um usuário recebe 6 segundos antes que uma instância expire. Isso é econômico e funciona como um recurso de segurança: quanto menos tempo uma instância tem, menos tempo um invasor tem para realizar um ataque. No entanto, construir uma aplicação com vários recursos, usar código não otimizado e adicionar camadas como recursos de segurança ao código exige o aumento dos tempos limite de função, o que pode dar a um agente mal-intencionado mais tempo para trabalhar e comprometer um app.

Embora esteja longe de ser uma solução completa de cibersegurança, como no caso de ataques de injeção maliciosos, um WAF ainda pode fornecer filtragem adicional para dados que estão sendo passados pelas funções do AWS Lambda como uma medida de segurança extra. Em geral, um WAF pode fornecer boa segurança com base em uma lista de condições de solicitação da web não permitidas, como a presença de código SQL ou um script que provavelmente seja malicioso e uma lista que permite a entrada de dados de fontes confiáveis.

Os usuários também podem se beneficiar de soluções que monitoram de perto as funções do AWS Lambda para garantir que as funções não sejam executadas com privilégios de administrador e não sejam expostas ao público, e que o rastreamento esteja habilitado para funções.[40]

Amazon API Gateway

Embora os ambientes de TI variem em tamanho e necessidades, a Amazon fornece as seguintes recomendações de segurança para usuários do Amazon API Gateway [41:]

- Implemente a política de privilégios mínimos ao criar, ler, atualizar ou excluir APIs.
- Monitore APIs REST e registre solicitações de API usando o Amazon CloudWatch[42] e o Amazon Kinesis Data Firehose[43].
- Habilite o AWS CloudTrail, um serviço da AWS que mantém um registro detalhado das ações tomadas por um usuário, uma função ou outro serviço da AWS no Amazon API Gateway para melhor monitoramento[44].
- Habilite o AWS Config, que permite aos usuários ver a configuração de todos os recursos da AWS em seus sistemas, verificar como os recursos estão conectados e ver a configuração história.[45]

Além disso, os usuários podem ativar o AWS WAF[46], que se integra com o Amazon API Gateway, ou usar soluções de terceiros para proteger as APIs do Amazon API Gateway de explorações comuns da web. As ferramentas de auditoria podem ajudar a garantir que as melhores práticas para o Amazon API Gateway sejam seguidas (por exemplo, que as APIs tenham a codificação de conteúdo habilitada).[47]

Tecnologia Serverless e Responsabilidade Compartilhada

tNo modelo serverless, os CSPs são responsáveis por proteger as infraestruturas críticas de software e hardware. No modelo de responsabilidade compartilhada, isso é conhecido como “segurança da nuvem”. Os CSPs também lidam com a segurança de outros componentes críticos, como criptografia do lado do servidor, sistema operacional e configuração de rede e firewall. Esses são componentes que fazem parte da “segurança na nuvem”, que em outros modelos de nuvem é de responsabilidade dos usuários. Os CSPs também implementam a política de privilégios mínimos e a abordagem de negação padrão para comunicações de serviço. No caso do AWS Lambda, a AWS cuida da infraestrutura subjacente, do sistema operacional e da plataforma da aplicação, enquanto os próprios usuários são responsáveis pela segurança de seu código, do armazenamento e da acessibilidade a dados confidenciais e do IAM em relação ao AWS Lambda e dentro de sua função.

Mas tudo isso não deve tornar os adotantes de serviços serverless complacentes com a segurança. Manter as configurações adequadas e a qualidade do código é fundamental para manter esses serviços o mais seguro possível. Mostramos neste artigo de pesquisa como agentes mal-intencionados podem tirar proveito de configurações incorretas e erros para lançar ataques em implantações serverless. Por causa de erros comuns do usuário, os agentes mal-intencionados nem precisariam fazer muito para realizar ataques. Em alguns casos, eles podem simplesmente consultar um bucket do Amazon S3 que hospeda conteúdo dinâmico (em vez de estático) para ver dados confidenciais e partes críticas do código. Configurações incorretas e políticas permissivas também conseguem permitir que ocorra o aumento de privilégios, permitindo que atacantes alterem permissões em políticas ou até mesmo obtenham privilégios de administrador.

Além de brechas de segurança que surgem de negligência ou descuido, agentes mal-intencionados também podem tirar proveito dos recursos de serviços serverless. Por

exemplo, as warm starts em chamadas de função, destinadas a reduzir a latência, podem ser utilizadas por atacantes que têm como objetivo lançar ataques furtivos e persistentes. Essas pessoas também podem abusar da escalabilidade elástica e automatizada oferecida por serviços serverless introduzindo código que pode consultar ininterruptamente as funções do AWS Lambda, esgotando assim os recursos financeiros das empresas.

A segurança dos serviços serverless pode ser ainda mais aprimorada com a implementação de certas medidas. Esses tipos de serviços conectados podem se beneficiar dos recursos de segurança, como uma opção de registro padrão aprimorada para Amazon S3 e uma configuração de autenticação padrão para endpoints do Amazon API Gateway. Os usuários fariam bem em seguir nossas recomendações de segurança para defender as implantações serverless de ameaças e riscos. Os desenvolvedores de soluções de segurança, por sua vez, devem considerar nossa orientação sobre os recursos vitais que as soluções de segurança devem ter para proteger adequadamente as implantações serverless de vulnerabilidades, hacks, configurações incorretas e outras ameaças.

Embora seja verdade que os componentes de computação de infraestrutura de serviços serverless sejam cobertos por CSPs, isso não significa que os usuários dos serviços não estejam sujeitos a fazer configurações incorretas, seja por negligência ou falta de conhecimento. Eles precisam entender que o modelo de responsabilidade compartilhada também se aplica à tecnologia. Os usuários de serviços serverless são responsáveis por manter os componentes críticos desses serviços tão seguros quanto possível, seguindo práticas de codificação seguras, mantendo os segredos protegidos, monitorando e registrando funções, configurando políticas de maneira adequada e implementando os princípios de privilégio mínimo e negação padrão. Ao detalhar os possíveis cenários de comprometimento e ataque entre os serviços conectados em uma arquitetura serverless, pretendemos destacar a importância da diligência na aplicação de práticas de segurança para impedir ameaças e mitigar riscos.

Referências

<?> Trend Micro. (Oct.24, 2019). *Trend Micro Security News.* "The Cloud: What it is and what it's for." Accessed on May 25, 2020, at https://www.trendmicro.com/vinfo/us/security/news/security-technology/the-cloud-what-it-is-and-what-it-s-for.

<?> Amazon Web Services. (March 22, 2019). *YouTube.* "Build a Serverless Startup in Just 30 Minutes!" Accessed on May 25, 2020, at https://www.youtube.com/watch?v=qBNYmYRITpU.

<?> Mark Nunnikhoven. (Oct. 22, 2019). *Trend Micro Simply Security.* "The Shared Responsibility Model." Accessed on May 25, 2020, at https://blog.trendmicro.com/the-shared-responsibility-model/.

<?> AWS. (n.d.). *AWS.* "Shared Responsibility Model." Accessed on May 25, 2020, at https://aws.amazon.com/compliance/shared-responsibility-model/.

<?> AWS. (n.d.). *AWS.* "The Shared Responsibility Model." Accessed on May 25, 2020, at https://docs.aws.amazon.com/whitepapers/latest/security-overview-aws-lambda/the-shared-responsibility-model.html.

<?> Security Best Practices in IAM. (n.d.). *AWS.* "Security Best Practices in IAM." Accessed on May 25, 2020, at https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#grant-least-privilege.

<?> Chris Munns. (July 23, 2018). *AWS Compute Blog.* "Powering HIPAA-compliant workloads using AWS Serverless technologies." Accessed on June 8, 2020, at https://aws.amazon.com/blogs/compute/powering-hipaa-compliant-workloads-using-aws-serverless-technologies/.

<?> Ihor Lobastov. (March 8, 2019). *DZone.* "Comparing Serverless Architecture Providers: AWS, Azure, Google, IBM, and Other FaaS Vendors." Accessed on June 3, 2020, at https://dzone.com/articles/comparing-serverless-architecture-providers-aws-az.

<?> David Ramel. (May 8, 2020). *Virtualization and Cloud Review.* "Cloud-Native Development Survey Details Kubernetes, Serverless Data." Accessed on May 25, 2020, at https://virtualizationreview.com/articles/2020/05/08/cloud-native-dev-survey.aspx.

<?> AWS. (n.d.). *AWS.* "Amazon S3." Accessed on May 25, 2020, at https://aws.amazon.com/s3/.

<?> AWS. (n.d.). *AWS.* "Using Amazon S3 block public access." Accessed on July 23, 2020, at https://docs.aws.amazon.com/AmazonS3/latest/dev/access-control-block-public-access.html.

<?> AWS. (n.d.). *AWS.* "Security Best Practices for Amazon S3." Accessed on May 25, 2020, at https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html.

<?> AWS. (n.d.). *AWS.* "How do I enable object-level logging for an S3 bucket with AWS CloudTrail data events?" Accessed on July 30, 2020, at https://docs.aws.amazon.com/AmazonS3/latest/user-guide/enable-cloudtrail-events.html.

<?> AWS. (n.d.). *AWS.* "Amazon Athena." Accessed on June 8, 2020, at https://aws.amazon.com/athena/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc.

<?> Morton Swimmer et al. (April 8, 2020). *Trend Micro Security News.* "Exploring Common Threats to Cloud Security." Accessed on May 25, 2020, at https://www.trendmicro.com/vinfo/us/security/news/virtualization-and-cloud/exploring-common-threats-to-cloud-security.

<?> Trend Micro Cloud Conformity. (n.d.). *Trend Micro Cloud Conformity.* "AWS S3 Best Practices." Accessed on May 25, 2020, at https://www.cloudconformity.com/knowledge-base/aws/S3/.

<?> AWS. (n.d.). *AWS.* "AWS Lambda Limits." Accessed on May 25, 2020, at https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html.

<?> Trend Micro Cloud Conformity. (n.d.). *Trend Micro Cloud Conformity.* "Cloud Conformity Lambda." Accessed on May 25, 2020, at https://www.cloudconformity.com/knowledge-base/aws/Lambda/.

<?> AWS. (n.d.). *AWS.* "Security best practices in Amazon API Gateway." Accessed on May 25, 2020, at https://docs.aws.amazon.com/apigateway/latest/developerguide/security-best-practices.html.

<?> AWS. (n.d.). *AWS.* "Amazon Cloudwatch." Accessed on May 25, 2020, at https://aws.amazon.com/cloudwatch/.

<?> AWS. (n.d.). *AWS.* "Amazon Kinesis Data Firehose." Accessed on May 25, 2020, at https://aws.amazon.com/kinesis/data-firehose/.

<?> AWS. (n.d.). *AWS.* "AWS CloudTrail." Accessed on May 25, 2020, at https://aws.amazon.com/cloudtrail/.

<?> AWS. (n.d.). *AWS.* "AWS Config." Accessed on May 25, 2020, at https://aws.amazon.com/config/.

<?> AWS. (n.d.). *AWS.* "AWS WAF - Web Application Firewall." Accessed on Aug. 11, 2020, at https://aws.amazon.com/waf/.

<?> Trend Micro Cloud Conformity. (n.d.). *Trend Micro Cloud Conformity.* "Amazon API Gateway Best Practices." Accessed on May 25, 2020, at https://www.cloudconformity.com/knowledge-base/aws/APIGateway/.

<?> AWS. (n.d.). *AWS.* “Security Best Practices in IAM.” Accessed on May 25, 2020, at https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html.

<?> Trend Micro Cloud Conformity. (n.d.). *Trend Micro Cloud Conformity.* “AWS IAM Best Practices.” Accessed on May 25, 2020, at https://www.cloudconformity.com/knowledge-base/aws/IAM/.

<?> Trend Micro. (n.d.). *Trend Micro.* “Trend Micro Cloud One Application Security.” Accessed on May 25, 2020, at https://www.trendmicro.com/en_ph/business/products/hybrid-cloud/cloud-one-application-security.html.

TREND MICRO™ RESEARCH

Trend Micro, líder global em cibersegurança, ajuda a tornar o mundo seguro para a troca de informações digitais.

A Trend Micro Research é desenvolvida por especialistas apaixonados por descobrir novas ameaças, compartilhar percepções importantes e apoiar os esforços para impedir os cibercriminosos. Nossa equipe global ajuda a identificar milhões de ameaças diariamente, lidera o setor em divulgações de vulnerabilidades e publica pesquisas inovadoras sobre novas técnicas de ameaças. Trabalhamos continuamente para antecipar novas ameaças e fornecer pesquisas instigantes.

www.trendmicro.com

TREND MICRO™ | research