eMag

InfoQ BRASIL

ONDE ESTAMOS?

Um panorama sobre o mercado de IoT frente ao surgimento de milhões de dispositivos programáveis.

SEGURANÇA

Um novo mundo onde falhas de segurança podem levar a cenários trágicos. Quais são os desafios?

BOAS PRÁTICAS

Especialistas na área compartilham boas práticas de desenvolvimento para soluções IoT

EMAG | **IOT**

APRESENTAÇÃO

O avanço da Internet das Coisas (IoT) trouxe para o cenário da tecnologia números e oportunidades até então inimagináveis. Estimativas dizem que até 2020 devemos ter, em todo o mundo, cerca de 25 bilhões de dispositivos conectados, além dos 4 bilhões de usuários. Serão cerca de 30 bilhões de produtores e consumidores de dados, trafegando informações a todo o tempo, sem parar.

Se por um lado IoT traz junto de si uma série de oportunidades, por outro lado traz também uma série de desafios e barreiras a serem vencidas no que diz respeito ao desenvolvimento de software. Essa eMag traz um compilado de conteúdos publicados no InfoQ Brasil que permite aos nosso leitores se ambientarem, para aqueles que ainda não são tão familiarizados com o assunto, e também acompanhar os desdobramentos de tamanho avanço tecnológico.

Iniciando nossa revista eletrônica, **Antero Taivalsaari** - da Nokia - e **Tommi Mikkonen** - da Universidade de Helsinki e da Mozilla Connected Devices - apresentam um roadmap para o mundo programável que IoT nos propõe. Em discussão, pontos como limitações atuais, desafios computacionais e tendências.

Logo em seguida, trouxemos um conjunto de 3 artigos discutindo um assunto de extrema relevância para o sucesso de soluções IoT: segurança. **Gary Evans**, experiente profissional do mercado do desenvolvimento de software, traz à tona questões relacionadas à segurança de soluções que sempre tiveram em questão para soluções tradicionais, mas que ficam ainda mais potencializadas em contextos envolvendo IoT. **Alasdair Allan**, aborda os reais impactos de problemas de segurança em soluções IoT e quais são as diferenças no que diz a respeito à segurança para uma solução tradicional, e uma solução IoT. Finalizando esse bloco, **Philipp Jovanovic** transforma toda sua experiência, por ter trabalhado com algoritmos de encriptação autenticada (NORX), em recomendações para desenvolvedores.

Trazendo o conteúdo para uma parte mais prática, **Wellington Mariusso** - CTO da Konker - traduz parte de sua experiência na construção de uma plataforma IoT em um texto sobre os desafios de se implementar serviços para dispositivos com recursos limitados. Encerrando o conteúdo dessa eMag, **Steven Willmott** - Diretor Sênior e Head de Infraestrutura de API Red Hat - parte de uma reflexão sobre como APIs e os recentes avanços tecnológicos estão contribuindo para o mundo programável, compartilhando as melhores práticas para se ter sucesso nessa área.

Esperamos que você tenha uma boa leitura e que esse material possa lhe auxiliar no entendimento e na caminhada dentro do mundo IoT.

CRIE SOLUÇÕES PARA INTERNET DAS COISAS COM A PLATAFORMA KONKER

A Plataforma Konker facilita a coleta, o gerenciamento e o processamento de dados dos seus dispositivos e sensores IoT. Construa rapidamente soluções conectadas para Internet das Coisas.

OS BENEFÍCIOS

PLATAFORMA OPEN SOURCE

Plataforma aberta para você construir, lançar e gerenciar a sua solução IoT

OTIMIZAÇÃO DE RECURSOS

Invista os seus recursos na construção do produto e não na infraestrutura

EXEMPLOS DE CÓDIGO

Exemplos pra você conectar seus dispositivos mais rápido

DISPOSITIVOS CONECTADOS

Com configurações simples usando nossos exemplos e APIs, fica muito mais fácil conectar seus dispositivos

DIFERENTES INTERFACES

Seja REST ou MQTT. tenha mais flexibilidade com as principais interfaces para IoT.

SAIBA MAIS EM
www.konkerlabs.com

konker

LEIA ONLINE

UM ROTEIRO PARA O MUNDO PROGRAMÁVEL

por Antero Taivalsaari e Tommi Mikkonen, traduzido por Eder Ignatowicz

Este arquivo foi originalmente publicado na revista IEEE Software. A IEEE Software oferece informação sólida e revisada por pares sobre os os recentes desafios estratégicos em tecnologia. Para enfrentar os desafios de uma execução confiável e flexível nas empresas, gerentes e líderes de TI confiam na soluções de última geração apresentadas na IT Pro.

O surgimento de milhões de dispositivos programáveis de forma remota em nosso dia a dia criará desafios significativos para os desenvolvedores de software. Contudo, ainda não foi desenvolvido um roteiro que aborde estes desafios a partir da perspectiva centrada em dados e nuvem dos sistemas atuais de IoT e em direção ao novo mundo programável.

A Internet das Coisas (IoT) representa o próximo passo significativo na evolução da internet. Na década de 70 e 80, a Internet era basicamente uma forma de conectar computadores. Os anos 90 e 2000, marcaram a Internet como a ferramenta de conexão entre as pessoas. Agora, a ênfase está mudando de forma a conectar tudo (todas as coisas) a Internet.

Grande parte da pesquisa atual de IoT é destinada ao problemas de aquisição de dados, análise destes dados em tempo real e offline, aprendizado de máquina, visualização de dados e outros tópicos importantes para big data1. Este enfoque não nos surpreende, dado ao enorme potencial de negócio que emerge da habilidade de coletar dados de milhões de dispositivos e sensores, processá-los e combiná-los de forma a fornecer novas informações sobre o comportamento da população e diversos outros fenômenos do mundo real.

Contudo, uma disrupção igualmente importante, porém mais sutil, está acontecendo. A evolução do hardware e o fácil acesso a chips integrados poderosos e de baixo custo, possibilita embarcar conectividade, máquinas virtuais e ambientes de execução de linguagens dinâmicas em todos os dispositivos. Desta forma, os dispositivos do nosso dia a dia como lâmpadas, maçanetas, sistemas de ar condicionado, regadores de jardim, aspiradores, escova de dentes e a pia da cozinha se tornarão conectadas e programáveis de forma dinâmica.

Neste artigo, nós apresentamos um roteiro a partir da perspectiva centrada em dados e nuvem dos sistemas atuais de IoT para um mundo onde os objetos do nosso dia dia estão conectados e a "ponta" das redes (network edge) é realmente programável. Este novo mundo programável trará novos desafios aos desenvolvedores de software. Os métodos mais populares de desenvolvimento, suas linguagens e ferramentas não são adequadas para o surgimento de milhões de dispositivos programáveis em nossas vidas. Estes desafios não estão entre os tópicos que recebem mais

atenção da comunidade de IoT, contudo, acreditamos que é necessário um estudo aprofundado dos problemas e desafios técnicos relacionados a esta realidade.

Como este artigo é baseado numa realidade futura, ele é de certa forma subjetivo. Nosso ponto de vista é baseado na experiência de nossos próprios projetos, nosso trabalho com IoT2-5, bem como da nossa experiência na previsão e participação nos últimos 20 anos de evolução da computação web e mobile. Por exemplo, no final dos anos 90, o surgimento de máquinas virtuais em dispositivos móveis não se apresentava como uma revolução tecnológica. Contudo, ela abriu o mercado de software em dispositivos móveis ao mercado de desenvolvedores e criou a hoje multibilionária indústria de aplicativos móveis. Embora a história raramente se repita, muitas vezes ela rima. Neste caso, um paralelo com o passado parece óbvio, porque estamos apenas chegando ao ponto em que poderemos embarcar programação dinâmica praticamente em todos os lugares e coisas.

O surgimento de uma arquitetura IoT comum de ponta a ponta

O termo "Internet of Things" (Internet das Coisas) não é novo. Nos últimos 20 anos, professores do MIT descrevem um mundo onde as coisas (dispositivos ou sensores) estão conectados e compartilham dados6. No princípio, os conceitos de IoT surgiram baseados em tecnologias como RFID e redes de sensores wireless7. Contudo, recentemente IoT se expandiu para vários domínios, incluindo saúde, transportes, energia, varejo e automação de processos. Centenas de startups foram criadas nestas áreas e a maioria destas empresas anunciaram componentes e uma plataforma de IoT. Um recente estudo listou 115 plataformas de cloud IoT8. Atualmente, nós estamos em um período onde a plataforma vencedora ainda não foi definida9. Provavelmente em meados de 2025, existirá apenas um pequeno, mas dominante, conjunto de fornecedores e plataformas de IoT.

O que é interessante em relação a grande gama de plataformas de IoT disponíveis é sua similaridade conceitual. Apesar de uma aparente diversidade e um grande número de fornecedores focados no mercado de IoT, uma arquitetura end-to-end comum está emergindo das soluções de IoT, pois um grande número de elementos são comuns a todos os sistemas1,10,11. A Figura 1 apresenta os conceitos básicos, que serão sumarizados a seguir .

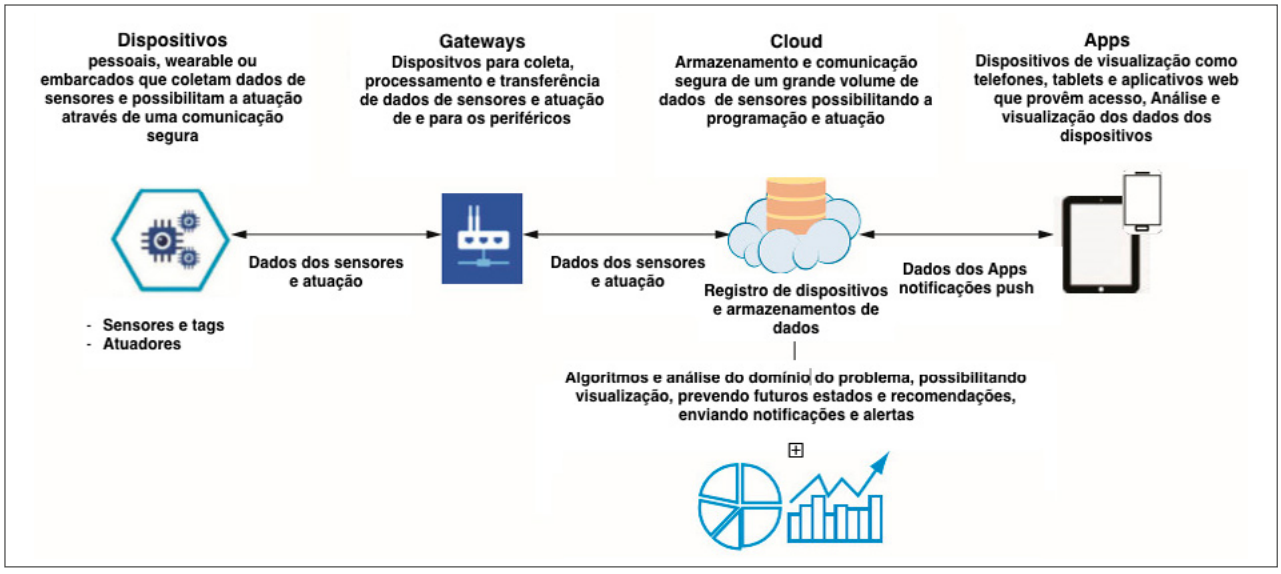


Figura 1 - O surgimento de uma arquitetura IoT comum end-to-end. Apesar de uma aparente diversidade e um grande número de fornecedores focados no mercado de IoT, uma arquitetura end-to-end comum está emergindo das soluções de IoT, pois um grande número de elementos são comuns a todos os sistemas

Dispositivos (devices)

Dispositivos (ou periféricos) são elementos de hardware que coletam dados de sensores ou realizam determinadas ações e que possuem capacidade de se comunicar de forma a transmitir os dados coletados ao ecossistema de IoT. Funcionalmente, dispositivos são sensores ou atuadores (actuators, que realizam ação).

Os sensores fornecem informações sobre a entidade física que monitoram. Esta informação pode variar desde a identidade da entidade física até variáveis mensuráveis, tais como temperatura, umidade, pressão, luminosidade, nível de som, fluxo de fluido, vibração e abrasão. Sensores cujo único propósito é facilitar um processo de identificação são chamados de tags.

Os atuadores utilizam energia, geralmente transportada por ar, eletricidade ou líquido, e a convertem em uma mudança de estado, modificando assim uma ou mais entidades físicas.

Gateways

Os gateways (ou hubs) coletam, pré-processam e transferem dados de dispositivos IoT e seus sensores, empregando diferentes protocolos de comunicação (geralmente sem fio), como Wi-Fi ou Bluetooth Smart. Os gateways fornecem tradução segura de protocolo entre dispositivos e a nuvem, e podem suportar tarefas como armazenamento e pré-processamento de dados, descoberta de serviços, geolocalização, verificação e cobrança. Os gateways também entregam os pedidos de atuação da nuvem para os dispositivos.

Em alguns sistemas, os próprios dispositivos podem carregar dados de detecção diretamente na nuvem e receber solicitações de atuação diretamente da nuvem - por exemplo, através de redes Wi-Fi, 3G ou 4G ou (em breve) NB-IoT (NarrowBand IoT) e 5G. Essas soluções não exigem gateways dedicados. Além disso, os próprios dispositivos geralmente atuam como gateways para outros dispositivos, possivelmente formando topologias de rede peer-to-peer ou mesh através da conectividade da rede local.

A Nuvem

Cloud computing (computação em nuvem) e as soluções de armazenamento e análise baseadas em nuvem são fundamentais para a maioria das plataformas IoT. Na arquitetura genérica de end-to-end (ponta a ponta), a nuvem desempenha três papéis principais.

O primeiro é a aquisição de dados, armazenamento e acesso. Uma funcionalidade fundamental nos sistemas IoT é a coleta e armazenamento de dados gerados pelos sensores. Os sensores de IoT geralmente coletam um grande volume de dados e estes devem ser armazenados na nuvem para eventual processamento e análise. As soluções nesta área vão desde bancos de dados simples a clusters de armazenamento amplamente replicáveis, tolerantes a falhas e escaláveis. As APIs de consulta e notificação para acessar os dados coletados também são fornecidas.

O segundo papel é a análise de dados. Isso se refere ao exame, conexão e transformação de dados proveniente de sensores de forma a descobrir e apresentar informações úteis, como por exemplo, para prover o compartilhamento de informações remotas e auxiliar na tomada de decisões. A análise em tempo real refere-se a funções de análise que são executadas imediatamente após o recebimento dos dados. Análise off-line se referem a operações em lote (batch) que ocorrem após o armazenamento de um grande conjunto de dados. Tecnologias de algoritmo, aprendizado de máquina e mineração de dados são importantes nesta área.

O terceiro papel é o suporte de atuação. Os fluxos de dados do sistema IoT não são apenas unidirecionais. Além da coleta de dados do sensor de dispositivos para a nuvem, é importante a capacidade de atuação segura do dispositivo através da nuvem. As capacidades de atuação são um facilitador fundamental para a programação remota do dispositivo.

Além disso, uma solução em nuvem que suporta o IoT geralmente inclui funções administrativas, como gerenciamento de dispositivos, gerenciamento de contas de usuários, registro de uso, monitoramento de status do servidor e recursos de relatórios.

O Roteiro

Acreditamos que o futuro da IoT reside na capacidade de orquestrar e programar de forma remota as grandes e complexas topologias dos dispositivos de IoT. Como argumentou Bill Wasik, uma vez que os dispositivos estão conectados a nuvens públicas ou privadas e que o fluxo de dados dos sensores e a capacidade de atuação nestes dispositivos esteja amplamente disponível, o foco passará da coleta e análise de dados dos sensores para a possibilidade de se programar aplicativos que manipulam sistemas complexos do mundo real12. Os recursos de atuação oferecidos pelos dispositivos IoT formam a fundação para toda esta mudança, pois nos permitem comandar e controlar os objetos em nossos arredores (e potencialmente em todo o planeta) a partir do conforto de um ambiente de programação e de nossos aplicativos.

Como mencionamos anteriormente, os avanços de hardware direcionarão a transição para o mundo programável. O avanço tecnológico do hardware e o custo de dispositivos de IoT estão rapidamente chegando a um ponto de inflexão no qual em breve poderemos executar sistemas operacionais completos, como por exemplo o Linux, stacks de software de forma virtualizada e até runtimes de linguagens dinâmicas em quase qualquer tipo de dispositivo. Os chips de baixo custo, que recentemente se tornaram disponíveis, já são iguais ou ou superiores em relação a memória e poder de processamento dos telefones celulares no final da década de 1990 - época do surgimento da plataforma Java 2, a Micro Edition13, marco inicial da revolução do aplicativo móvel. Desta forma, acreditamos que em breve estarão disponíveis em nossos dispositivos IoT um ambiente de programação multi-plataforma similares ao Java 2.

Outra tendência importante que impulsiona a indústria em relação ao mundo programável é a edge computing (computação de fronteira ou borda). Os sistemas clássicos

de computação em nuvem são altamente centralizados. Embora a computação centralizada tenha benefícios significativos, pode ser dispendiosa em termos de comunicação e consumo de energia. Considere um ambiente IoT consistindo de uma grande coleção de dispositivos próximos uns dos outros. Pode ser ineficiente transmitir os dados a serem processados desses dispositivos para datacenter remoto e, em seguida, transmitir os pedidos de atuação de volta do datacenter para os dispositivos. Em uma implantação IoT com requisitos de latência restritos, a sobrecarga de latência por si só pode tornar tais soluções impraticáveis.

O termo "edge computing" (computação de fronteira) foi cunhado em 2002 e foi originalmente associado à implantação de aplicativos em redes de entrega de conteúdo (CDNs). O objetivo principal era se beneficiar da proximidade de servidores de borda CDN para melhorar a escalabilidade e menor latência. As soluções IoT de computação de borda utilizam a borda da rede (geralmente dispositivos de gateway, como roteadores ou estações base) para computação, por exemplo, para pré-processar dados de sensores, disparar alertas e solicitações de atuação localmente com base em critérios pré-definidos. Tais sistemas podem beneficiar de tecnologias de conectividade locais (por exemplo, Bluetooth Smart ou Wi-Fi Direct) para permitir a comunicação direta, de forma eficiente e descentralizada entre os dispositivos sensores. As tecnologias de virtualização também desempenham um papel central na ativação da migração de computação entre dispositivos.

A Tabela 1 apresenta nosso roteiro que prevê como os sistemas IoT evoluirão nos próximos 10 anos, tanto do pontos de vista de dados quanto de programação. A tabela é baseada em:

- Tendências observadas na indústria e na academia, refletindo teses recentes, trabalhos acadêmicos e livros publicados 1, 5, 7, 10, 11, 14, 15;
- Opiniões de especialistas baseadas em experiências práticas de desenvolvimento de produtos e protótipos tanto na indústria quanto na academia 2, 3;
- Experiência pessoal e observações anteriores proveniente da revolução dos mobileapps, à medida que os telefones celulares evoluíram de dispositivos fechados e centrados na voz para smartphones de aplicativos 13;
- Experiência pessoal e observações anteriores da evolução da web, a partir de um ambiente de distribuição de documentos simples para uma plataforma que suporta um rico desenvolvimento de aplicativos e um deploys em escala mundial de forma instantânea 16.

Year	Panorama dos Dados	Panorama da Programabilidade
2015	• Aquisição de dados em larga escala; • Monitoramento, rastreamento, roteamento, comando e controle e mineração de tendências e comportamento; • Análise dos dados centrada na cloud, incluindo análise em tempo real e off-line; • Ênfase na visualização de dados e alertas simples; • Tecnologias open source disponíveis e amplamente utilizadas para análise e aquisição dos dados;	• Grande parte da computação processada em cloud; • Suporte básico a atuação em dispositivos, geralmente implementada de forma nativa ou de uma API e Apps proprietários; • Aplicações de controle de dispositivos proprietárias e exclusivas de um dispositivo ou fornecedor; • Surgimento de notações visuais (como o Node-RED) para uma portabilidade maior entre dispositivos; • Surgimento de padrões (como o OMA Lightweight M2M e IPSO SMART Objects)
2020 (Era do Edge IOT)	• APIS de computação e mecanismos de fronteiras dependem extensivamente no processamento e análise de dados, filtros mais inteligentes; • Aumento da operação, aquisição e análise de dados de forma autônoma baseada em computação máquina a máquina através de comunicação local • Adaptação, expansão e aprimoramento automático através da seleção dos recursos de computação (cloud x edge)	• Computação de fronteira e APIs disponíveis para provisionamento da computação entre dispositivos; • Capacidade mais avançada de atuação, APIs de atuação padrão; • Aplicações padrão de controle de dispositivos específicos de um determinado domínio, como exemplo no controle de iluminação e segurança das casas; • Máquinas virtuais disponíveis em grande parte dos dispositivos, possibilitando aplicações que funcionem em múltiplos fornecedores e migração flexível da computação entre a cloud e dispositivos de borda.
2025 (Era da IoT Universal)	• Aquisição de dados automática e baseada no contexto, otimização das decisões através do uso de técnicas de inteligência artificial como machine learning; • Colaboração universal máquina-máquina através de APIs comuns; • Compartilhamento de dados entre as máquinas.	• Um modelo de desenvolvimento e execução de aplicações universal e conteinerizada suportando múltiplos fabricantes e indústrias; • APIs genéricas a toda indústria provendo descoberta, aquisição de dados, programação remota de dispositivos e gerência; • Possibilidade da criação de consoles universais; • Programação remota dinâmica de dispositivos, possibilitado pelo uso de máquinas virtuais, containers e APIs padrão

Basicamente, esperamos que os recursos de programação da IoT evoluam a partir de recursos de atuação simples, aplicativos específicos do fornecedor, APIs de dispositivos e aquisição e processamento de dados centrados na nuvem, para sistemas que ampliem a computação, virtualização e os contêineres na ponta das redes. Estes sistemas suportarão o desenvolvimento de aplicações portáveis e independentes de fornecedor e indústria. Eles também permitirão, quando necessário, a migração flexível de computação e dados entre a nuvem e dispositivos de borda heterogêneos. Nós denominamos as próximas fases da evolução dos sistemas IoT como a Era Edge IoT ou a Era do IoT Universal.

É discutível se somente uma uma API abrangerá dispositivos IoT de domínios totalmente diferentes e proveniente de mercados verticais. No entanto, é seguro prever que em 5 a 10 anos, os dispositivos IoT e suas API terão convergido significativamente. Também é impraticável que as pessoas sejam obrigadas a utilizar um app específico para cada fornecedor e para cada dispositivo. Além disso, é provável que a infraestrutura necessária para esta mudança cresça em torno da rede IP e da web, contudo, a web ainda necessita ser aprimorada de forma a suportar tecnologias de conectividade locais de forma a abranger a computação de ponta.

O que torna o desenvolvimento para IoT diferente

São sete os pontos que diferem desenvolvimento do IoT do desenvolvimento de aplicativos móveis e do desenvolvimento web client-side. Essas diferenças são fundamentais para as implicações e os desafios técnicos que serão apresentados.

Primeiro, os dispositivos IoT são quase sempre parte de um sistema maior de dispositivos, por exemplo, uma instalação de dispositivos interoperantes em uma casa, um escritório, uma fábrica, um shopping center ou um avião. Além disso, os dispositivos IoT geralmente são apenas uma pequena parte da arquitetura de end-to-end que discutimos anteriormente. É fato que, PCs e smartphones também possuem uma grande dependências em serviços baseados em nuvem. No entanto, do ponto de vista do desenvolvedor do aplicativo, eles ainda são principalmente dispositivos isolados (standalone), com o desenvolvedor programando para um único computador ou smartphone.

Em segundo lugar, os sistemas IOT nunca dormem. PCs, smartphones e outros dispositivos de computação autônomos podem ser vistos como sistemas "rebootables" (reiniciáveis) que podem e serão reiniciados quando as coisas irem mal. Em contraste, os sistemas IOT geralmente não devem ou não podem ser encerrados na sua totalidade. Embora os dispositivos individuais possam ser desligados, todo o sistema de IoT deve ser resiliente para interrupções de serviço nos dispositivos e redes.

Em terceiro lugar, sistemas de IoT são mais parecidos com um rebanho do que com um único animal de estimação. O número de unidades de computação (dispositivos ou CPUs) nos sistemas IoT usualmente é dramaticamente maior do que nos ambientes de computação tradicionais, potencialmente alcançando centenas, milhares ou mesmo milhões de dispositivos. Ao contrário das PCs e smartphones, sistemas em que os usuários os tratam quase como animais de estimação, os dispositivos de um grande sistema de IoT são mais parecidos com um rebanho, pois devem ser gerenciados em massa ao invés de receber atenção e cuidados individuais.

Em quarto lugar, dispositivos IoT são muitas vezes incorporados em nossos ambientes, de modo que sejam fisicamente invisíveis e inalcançáveis. Os dispositivos podem estar permanentemente no subsolo ou incorporados fisicamente em vários materiais (por exemplo, sensores de vibração em equipamentos de mineração). Pode ser impossível conectar cabos físicos a esses dispositivos ou substituir hardware ou software embarcado para tentar solucionar eventuais problemas.

Em quinto lugar, os sistemas IoT são extremamente heterogêneos. Suas unidades de computação podem variar drasticamente em relação a poder de computação, recursos de armazenamento, largura de banda de rede e requisitos de energia. Além disso, também podemos encontrar uma ampla gama de mecanismos de I/O, funcionalidades dos sensores e modalidades de input (entrada) disponíveis. Alguns dispositivos possuem botões físicos e visores, enquanto muitos dispositivos não possuem nenhuma interface de usuário visível.

O sexto ponto é que sistemas IoT tendem a possuir uma conectividade fraca, com conexões de rede intermitentes e muitas vezes pouco confiáveis. Do ponto de vista do desenvolvedor de software, esse ambiente impõe a necessidade de se preparar constantemente para a falha. As aplicações com um tratamento de erros precário provavelmente serão bloqueadas durante uma interrupção do dispositivo ou da rede, por exemplo esperando infinitamente por um pacote de resposta ou memória, energia ou qualquer outro recurso.

Por fim, as topologias do sistema IoT podem ser dinâmicas e efêmeras. Por exemplo, plantas de fábrica podem ter uma multiplicidade de peças de equipamentos constantemente em movimento com capacidades de detecção. Ou ainda elas podem produzir produtos (ou partes) que permanecem no perímetro da fábrica apenas de forma transitória. Estes exemplos, combinados com um grande número de dispositivos, exigirão tecnologias de implantação e pro-

gramação que possam lidar com a mudança dinâmica de "enxames" (swarms) de dispositivos.

Além dessas diferenças básicas, muitas questões adicionais surgem da relativa imaturidade do sistemas de IoT. Para ilustrar estas questões, detalharemos dois exemplos:

Primeiro, os sistemas IoT de hoje possuem APIs para desenvolvimento imaturas incompatíveis. APIs de desenvolvimento comuns a toda indústria ainda não foram desenvolvidas. Ao contrário do desenvolvimento de aplicativos para aplicativos móveis e web, em que já houve uma convergência significativa, As APIs da IoT ainda tendem a ser específicas para cada fornecedor e para cada hardware. Contudo, vários esforços de padronização estão em andamento, por exemplo Industrial Internet Consortium, IPSO Alliance, Open Connectivity Foundation (antigamente denominada Open Interconnect Consortium), e a Open Mobile Alliance. No entanto, estas iniciativas de padronização ainda levarão vários anos para alcançar o consenso e a maturidade.

Além disto, os sistemas IoT atuais estão centrados na nuvem, ou seja, quase todos os dados são coletados e armazenados na cloud e a maioria das computações ou ações nos dados coletados também ocorrem no lado da nuvem. No entanto, à medida que os dispositivos e gateways IoT se tornam mais robustos, a computação pode ocorrer em vários locais: dispositivos, gateways ou a nuvem. O comportamento ideal do sistema IoT dependerá da sua capacidade de migrar a computação e os dados de forma flexível para os dispositivos nos quais os cálculos fazem mais sentido em um determinado momento.

Implicações e Desafios para o Desenvolvimento de Software

Para resumir as diferenças que acabamos de apresentar, os desenvolvedores de IoT devem considerar várias dimensões que não são familiares para a maioria dos desenvolvedores de aplicativos para dispositivos móveis e web (client side), incluindo:

- Programação multi dispositivo;
- A natureza reativa e sempre online dos sistemas;
- Heterogeneidade e diversidade;
- A natureza distribuída, altamente dinâmica e potencialmente migratória do software; e
- O requisito primordial de se escrever software de forma defensiva e tolerante a falhas.

Uma aplicação IoT típica é contínua e reativa. Com base nas leituras observadas dos sensores, os cálculos são disparadas e eventualmente resultam em vários eventos que geram algum tipo de ação. Os programas são essencialmente assíncronos, paralelos e distribuídos.

Estritamente falando, essas características não são novas no desenvolvimento de software. Qualquer desenvolvedor que criou software para sistemas distribuídos ou de missão crítica é pelo menos um pouco familiarizado com os desafios decorrentes dessas características. Os desenvolvedores do software de backends de cloud distribuídos em cluster também enfrentam esses mesmos desafios.

Falácias da Computação Distribuída

No entanto, com base em nossa experiência, acreditamos que o desenvolvedor de aplicações Web e Mobile não estão preparados para enfrentar os desafios do desenvolvimento de sistemas de IoT. Como L. Peter Deutsch e James Gosling apropriadamente resumiram no seu artigo "Falácias da Computação Distribuída", invariavelmente, programadores possuem oito suposições falsas quando escrever software para sistemas distribuídos pela primeira vez 17:

- A rede é confiável;
- A latência é zero;
- A largura de banda é infinita;
- A rede é segura;
- A topologia não muda;
- Existe somente um administrador;
- O custo de transporte é zero.

A falta de consideração desses falsos pressupostos resultará em vários tipos de erros - por exemplo, sockets abertos que escutam dispositivos que não estão mais presentes, assumir respostas imediatas ou aguardar infinitamente pacotes de resposta, ou ainda consumindo memória e baterias desnecessariamente. Como Leslie Lamport escreveu, em um sistema distribuído, "o fracasso de um computador que você nem sabia que existe pode tornar seu próprio computador inutilizável" 18.

Dito isto, existe o perigo de que os desenvolvedores terão que escrever muito código para lidar com qualquer potencial erro e exceção. Isso poderia enterrar a lógica do programa real em milhares de linhas de código de controle de erros, tornando os programas muito mais difíceis de serem mantidos e compreendidos. Portanto, um grande desafio no desenvolvimento do IoT é alcançar o equilíbrio adequado entre a lógica da aplicação e o tratamento de erros. Idealmente, as linguagens e ferramentas de desenvolvimento devem auxiliar o desenvolvedor, de forma que o código de manipulação de erros não torne incompreensível a lógica do programa.

Em geral, os custos ocultos de construção e manutenção de software para sistemas distribuídos são quase sem-

pre subestimados. De acordo com estudos, as atividades de verificação, testes e validação podem representar até 75% dos custos totais de desenvolvimento19.

Linguagens e ferramentas inadequadas

As linguagens de programação e as ferramentas utilizadas para o desenvolvimento do IoT são, em grande parte, as mesmas usadas para o desenvolvimento de aplicativos mobile e web Por exemplo, os kits de ferramentas de desenvolvimento de software para as populares placas de desenvolvimento do IoT como Arduino, Espruino, Edison e Galileo da Intel e Tessel, oferecem a escolha entre as linguagens C, C #, Java, JavaScript ou Python.

Indo contra todas as probabilidades, JavaScript e Node.js (a versão do lado do servidor do JavaScript) estão se tornando ferramentas centrais para o desenvolvimento do IoT devido a sua popularidade no desenvolvimento web. Contudo, esta é uma escolha infeliz pois o JavaScript não foi projetado para escrever aplicativos assíncronos e distribuídos. Recentemente, a expressão "callback hell" tornou-se popular em programas escritos em JavaScript pois caracteriza situações nas quais a lógica da aplicação se torna impossível de entender, devido ao número de chamadas de função assíncronas e funções de gerenciamento de eventos utilizadas para a lidar com eventos de sucesso e manipulação de erros (Veja, por exemplo, Callback Hell).

As atuais linguagens populares de desenvolvimento IoT também não abordam os aspectos de programação em larga escala, pois elas não provêm mecanismos de orquestração de sistemas para milhares de dispositivos e nem permitiriam que o código migre com flexibilidade entre a nuvem, gateways e os dispositivos.

A natureza dinâmica dos sistemas de IoT

De forma geral, o desenvolvimento de software se tornou muito mais ágil e dinâmico nos últimos 10 a 15 anos. A maioria dos aplicativos está conectado a serviços back-end. Linguagens de programação dinâmicas, como JavaScript e Python, ganharam popularidade. O surgimento da World Wide Web permitiu o deploy instantânea de software de qualquer lugar do mundo. Os desenvolvedores esperam poder atualizar suas aplicações em um ritmo drasticamente mais rápido do que a 10 anos atrás - e muitas muitas vezes, várias atualizações por hora. O modelo de desenvolvimento e deploy do DevOps20 substituiu a maioria das práticas anteriores de software, automatizando a entrega de software e infraestrutura.

Embora esses avanços tenham beneficiado a indústria de software, a natureza extremamente dinâmica dos sistemas IoT apresenta desafios adicionais. Por exemplo, depurar (debugging) e testar sistemas IoT pode ser um grande desafio dado o grande número de dispositivos, as topologias dinâmicas, a conectividade não confiável e a heterogeneidade dos dispositivos.

Embora um dispositivo IoT individualmente possa ser facilmente testado pois possui funcionalidades limitada de coleta de dados, o teste de um sistema inteiro composto por centenas ou milhares de dispositivos implantados em ambientes complexos do mundo real (fábricas, shoppings, estufas, navios, etc.) pode ser desafiador. A depuração e o teste tornam-se ainda mais complicados se o sistema puder se ajustar e equilibrar (trocar) a velocidade de computação, a latência da rede e o consumo de bateria do dispositivo ao migrar dinamicamente a computação entre a nuvem, gateways e dispositivos. Nesse caso, o comportamento do sistema pode não ser totalmente replicável para testes e depuração.

Novamente, para os desenvolvedores de software distribuído ou de missão crítica ou sistemas de automação de processos, esses desafios não são necessariamente novos. No entanto, a grande maioria dos desenvolvedores de aplicações web e de aplicações para dispositivos móveis não enfrentaram tais desafios e, portanto, provavelmente subestimam o esforço e os potenciais problemas associados à depuração e teste destes sistemas.

Olhando para frente, mas atento às lições do passado

As observações e os desafios apresentados revelam que o surgimento do mundo programável exigirá muito mais do que apenas um novo desenvolvimento de hardware, novos protocolos e tecnologias de comunicação ou ainda novas técnicas para analisar enorme conjuntos de dados. Para aproveitar todo o poder do mundo programável, precisaremos de novas tecnologias, processos, metodologias, abstrações e ferramentas de engenharia e desenvolvimento de software. Experiências anteriores e tecnologias existentes para o desenvolvimento de sistemas distribuídos e software de missão crítica desempenham um papel importante na redução do trabalho duplicado e na reinvenção da roda. Em grande parte, os desafios precisam ser abordados educando os desenvolvedores de software para perceber que o desenvolvimento do IoT realmente difere do desenvolvimento de aplicativos para dispositivos móveis e de aplicativos do web do lado cliente.

Segurança

Um dos maiores desafios para a efetivação do mundo programável é a segurança. O gerenciamento remoto de instalações complexas de dispositivos IoT em ambientes como fábricas, usinas de energia ou plataformas de petróleo exige, obviamente, a máxima atenção à segurança. Os

recursos de atuação e programação remota podem representar riscos importantes de segurança. Protocolos de criptografia para a segurança da camada de transporte, certificados de segurança, isolamento físico e outras práticas industriais estabelecidas desempenham um papel fundamental nessa área, mas ainda existem vários desafios técnicos interessantes. No entanto, talvez na prática o risco de segurança mais significativo seja que milhares de dispositivos IoT ainda tenham suas configurações de segurança padrão, muitas vezes com a senha de administrador padrão. Na Finlândia, já ocorreram incidentes em que os sistemas de aquecimento de blocos de apartamentos inteiros foram desligados remotamente porque seus sistemas de controle foram deixados expostos a ataques baseados na Internet.

Discussão

Até o presente momento, os recursos de programação dos sistemas IoT são em sua maioria apresentados na forma de aplicativos personalizados de diferentes fabricantes para controlar equipamentos específicos de seus ecossistemas. Por exemplo, as lâmpadas Philips podem ser controladas com o aplicativo Philips Hue, enquanto a GE e o Cree fornecem aplicativos diferentes para seus sistemas de iluminação. Do mesmo modo, os sistemas de ar condicionado, sistemas de segurança, sistemas de controle de mídia doméstica e aplicativos de controle remoto de carro de diferentes fabricantes normalmente exigem aplicativos distintos. A abordagem "aplicação separada para cada coisa" não escala bem e é apenas uma solução temporária até que os padrões da indústria sejam definidos.

Nós prevemos nos próximos anos uma mudança na maneira que desenvolvemos soluções de IoT. Até o momento, não existem ambientes de desenvolvimento de software universalmente interoperáveis que permitam aos desenvolvedores escreverem sem esforço, um aplicativo IoT capaz de ser executado em todos os tipos de dispositivos, e muito menos existe uma forma de se orquestrar e gerenciar topologias grandes e complexas de instalações heterogêneas de tais dispositivos. A falta de tais ferramentas reflete a imaturidade e a fragmentação do mercado IoT, bem como as limitações técnicas dos dispositivos - por exemplo, CPUs com capacidade restrita ou a incapacidade de atualizar o software sem conectar cabos físicos aos dispositivos. As máquinas virtuais, os ambientes de execução de linguagens dinâmicas e as técnicas de software "líquido" desempenharão um papel fundamental tornando possível transferir dados e computação de forma flexível entre dispositivos. (Um software líquido funciona perfeitamente em vários dispositivos de propriedade de um ou mais usuários.) O software binário tradicional e os kits de desenvolvimento específicos de hardware IoT estão em desvantagem significativa se o mesmo código deve ser executado (ou adaptável à execução) em uma ampla gama de dispositivos.

O mundo programável apresenta emocionantes oportunidades e desafios. Esperamos que este artigo - e particularmente o nosso roteiro - inspire e incentive as pessoas a trabalharem arduamente nesta área.

Agradecimentos

A Academy of Finland (projeto 295913), Mozilla e Nokia Technologies apoiaram esta pesquisa

LEIA ONLINE

LEIA ONLINE

ESTE SOFTWARE É SEGURO?

por Gary K. Evans, traduzido por Eder Ignatowicz

Esse software é seguro? Em toda a minha carreira, ouvi essa pergunta de apenas um dos meus funcionários e meus clientes de consultoria.

Em nossa batalha de metodologias, nós gastamos uma energia imensa em funcionalidades, histórias, épicos, técnicas de extração de requisitos e metodologias de testes. Mas afinal, o que produzimos é seguro?

Um produto seguro:

- Não causa danos materiais, financeiros e nem aos dados;
- Não causa danos físicos, emocionais ou de reputação aos seres humanos ou outras criaturas;
- Resiste a ameaças de segurança de fontes maliciosas as quais podem comprometer as características #1 ou #2.

Estado atual da área

Em Julho de 2015, os white-hat (chapéu branco) hackers Charlie Miller e Chris Valasek assumiram o controle do Jeep Cherokee 2014 conduzido pelo jornalista Andy Greenberg. Enquanto Greenberg dirigia a 70 milhas por hora, Miller e Valasek, sentados em um local a 10 milhas do veículo, manipularam o ar condicionado do carro, o rádio, a exibição do painel, habilitaram e desabilitaram os freios e, por fim, causaram completa pane no veículo. Depois que a história de Greenberg ganhou repercussão nacional, a Fiat Chrysler realizou o recall de 1,4 milhão de Cherokees, para dois meses depois se envergonhar novamente quando teve que realizar outro recall de 8,000 SUVs equipadas com o mesmo componente inseguro UConnect.

Uma coletânea de histórias de hacking automotivo é mantida no site de Security Affairs (Aventuras/Casos de Segurança). Como engenheiro de software, estou profundamente incomodado com o fato de que especialistas da indústria afirmam que os fabricantes de automóveis estão muito defasados em relação ao resto da indústria de software. De acordo com Ed Adams, pesquisador da Inovação de Segurança que testa a segurança de automóveis:

> *As técnicas e tecnologias utilizadas por fabricantes de automóveis sao ultrapassadas. Os softwares de automóveis não são criados com os mesmos padrões, por exemplo, de um sistema bancário, ou do software que é produzido na Microsoft.*

Enquanto buscamos umaInternet das coisas onipresente, quando somos desafiados com uma nova e perturbadora vulnerabilidade nos questionamos: "Por que alguém não antecipou isso?" Sergey Lozhkin, da Kaspersky Lab, relatou falhas de segurança perturbadoras em equipamentos médicos. Lozhkin concentrou sua atenção nas clínicas para determinar a vulnerabilidade desses equipamentos.

Em uma das clínicas, ele relatou:

> *Testei a rede Wi-Fi e consegui me conectar a um equipamento de ressonância magnética (MRI), e encontrar informações pessoais e falhas na arquitetura. Fiquei assustado com a facilidade. O vetor inicial do ataque foi a rede Wi-Fi, pois a rede não era tão segura quanto deveria, especialmente em um lugar onde você mantém registros médicos (dados de saúde dos pacientes).*

Seus relatórios terminam com uma visão perturbadora. Lozhkin afirmou:

> *Esta é uma área onde os pesquisadores de segurança "white-hat" e criminosos têm muito interesse, pois trata-se de um grande mercado potencial. Como exemplos podemos citar a pirataria de carros, carros conectados, equipamentos médicos, entre outros.*

A Insegurança é sutil

O Juramento de Hipócrates diz: "Acima de tudo, não faça mal". Contudo, falhas de segurança muitas vezes podem ser sutis, por exemplo, uma mensagem enviada duas vezes devido a um atraso de rede ou ainda um commit corrompido em um banco de dados. Muitas vezes nós consideramos estes erros como simples e abstratos, mas a diferença entre um remédio e o um veneno é apenas na dosagem. Estas falhas podem levar seu banco de dados a um estado corrompido mas ainda operacional, gerando inconsistências posteriores difíceis de se rastrear.

Testar melhor nosso software é apenas parte da solução para tornar nossos códigos mais seguros. Uma mentalidade de verificação agressiva de segurança é essencial. Mas como determinar se nosso software possui vulnerabilidades?

Nem sempre é fácil encontrar esta resposta pois muitas vezes não conseguimos nem imaginar cenários de falha. Por exemplo, um editor de texto pode causar mal?

Considere que você criou um documento em seu editor de texto como o abaixo. O símbolo " <\t> " representa o caractere TAB.

Adam Jones	<\t>	<\t> **X**
Barbara Martin	<\t>	<\t> **X**
Chester Smith	<\t>	<\t> **X**

Você salva o documento e, depois de algum tempo, abre o mesmo arquivo no seu editor. Contudo, a estrutura do documento é diferente.

Adam Jones	<\t>	<\t> **X**
Barbara Martin	<\t>	<\t> **X**
Chester Smith	<\t> **X**	

Nós percebemos que um caractere TAB desapareceu. Mas quanto isto é perigoso? A resposta é o que foi omitido do trecho anterior. Foi retirado o cabeçalho das colunas correspondentes a cada TAB. Este é o documento "real" contendo os cabeçalhos: prisoner name (nome do prisioneiro), to be hanged (a ser enforcado) e to be released (para ser solto):

Prisoner Name	To Be Hanged	To Be Released
Adam Jones	<\t>	<\t> **X**
Barbara Martin	<\t>	<\t> **X**
Chester Smith	<\t> **X**	

"Me perdoe Chester, este bug será corrigido no próximo release." Este exemplo é originado de um Plano de Teste da Marinha dos EUA para fidelidade de documentos. O cenário foi alterado para simplicidade.

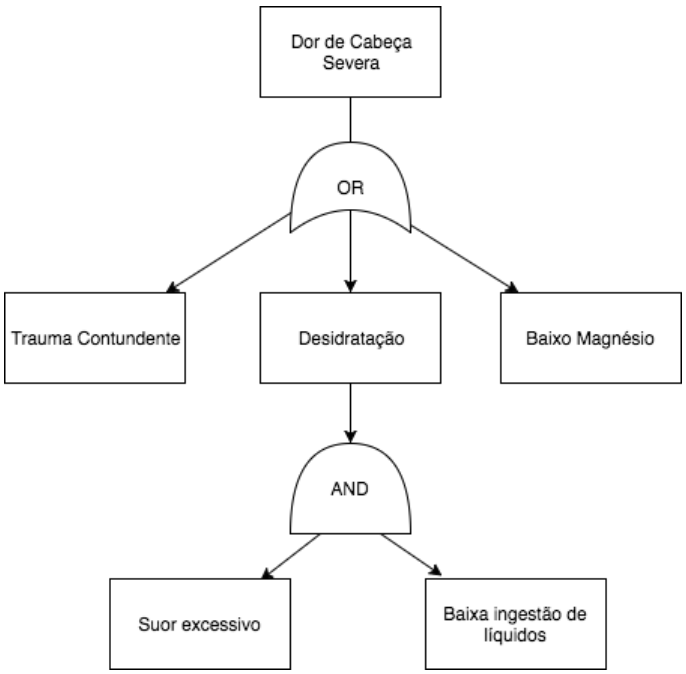
Soluções do mundo real

Para determinar o nível de segurança, nós não devemos somente confiar nos testes no nosso código do produto. Quantidade de defeitos e segurança são questões independentes. Zero defeitos pode ainda significar software inseguro pois defeitos são relacionados à especificação do sistema, mas ela muitas vezes pode estar incorreta. Esta é a razão para muitos defeitos nos surpreenderem: "Como isto aconteceu"", "Isto não deveria nunca ter acontecido" ou "Eu nunca pensei neste caso".

Uma técnica de análise importante que podemos utilizar no desenvolvimento dos produtos é a Fault Tree Analysis (FTA - Análise da Árvore de Falhas). A FTA produz um modelo visual de eventos e caminhos que podem levar a falha de um componente de sistema. Conceitualmente, a análise da árvore de falhas possibilita identificar estados finais/terminais não pretendidos ou inseguros e, ainda, identificar as condições que levaram a este estado terminal.

A FTA incorpora conceitos tradicionais de lógica booleana (AND, OR, XOR) como gates (portas). Caminhos e eventos alimentam estes gates e geram a árvore dos estados indesejáveis. O exemplo abaixo é dos estados de dor de cabeça severa e suas causas potenciais.

FTA é uma técnica poderosa desenvolvida para projeto de sistemas com estados finais conhecidos e é baseado na



filosofia que a falha destes sistemas é originada da falha dos componentes de hardware e software. Diagramas de árvore de falhas são simples de serem criados por qualquer profissional técnico, por exemplo, no Microsoft Visio. Baseado na minha experiência, FTAs combinados com diagramas UML de máquinas de estados são muito úteis para entender o que pode ocorrer de errado em um sistema.

Uma alternativa às FTAs é a técnica STAMP (System-Theoretic Accident Model and Processes). Criada pelo especialista em segurança Nancy Leveson, esta técnica foca no sistema como um todo e na interação entre os componentes do sistema. A tese de Levenson é que falhas ocorrem quando os sistemas entram em estado perigoso (hazardous) e este estado é consequência da aplicação inadequada das restrições de segurança ao comportamento do sistema.

Por se tratar de uma visão sistência, a técnica STAMP é mais complexa quando comparada a criar diagramas de análise da árvore de falhas. STAMP possui uma semântica rica e suporta diversas perspectivas na análise dos fatos, incluindo: cadeia de eventos, condições casuais e agentes que contribuíram para a condição de falha sistência. Por também incorporar restrições, níveis hierárquicos de controle e modelo de processos, esta técnica também pode ser aplicada a desenvolvimento e entrega de produtos.

E como isto se aplica ao desenvolvimento Web?

O desenvolvimento de software evoluiu de aplicações simples para o que eu chamo de "softwares" - interação entre múltiplas unidades funcionais, em diversas linguagens, executadas em diversas plataformas de forma concorrente e cada uma destas unidades é produzida por grupos e empresas diferentes. Nós vivemos em um mundo de sistemas de sistemas e a interação na interfaces destes produtos são os pontos de falha potencial. Devemos adotar todas as ferramentas que ajudem a termos uma visão mais clara da segurança (safety) do software que produzimos.

Preocupações relativas à segurança são presentes desde a especificação até o release do produto. Para enfrentar estas questões, nós não devemos esperar a resposta perfeita: se focada, mesmo uma abordagem fragmentada pode ser um importante passo em direção a segurança dos nossos sistemas.

Diagramas UML de máquinas de estado nos ajudam a identificar estados que são perigosos ou possam corromper o sistema. Um exemplo é o estado "catch all" (pegue tudo) que ocorre quando o programador não sabe qual lógica executar e enquanto um componente estiver neste estado inválido, um evento ou mensagem pode ser recebido e não será processado.

Avaliar cenários de múltiplas threads é sempre complicado. Nós devemos buscar uma análise compreensiva das regiões críticas, a possibilidade de condições de corrida (race conditions), deadlocks potenciais e outras anomalias de multithreading.

Quando seu código detecta um erro, ele deve encerrar a thread, o processo ou a aplicação? Ou ele deve continuar executando o sistema em modo limitado? Nós devemos listar os benefícios e riscos de cada um destes cenários e escolher o melhor entre eles, sempre tendo a segurança em primeiro lugar.

A complexidade das aplicações modernas é assombrosa e o acesso não autorizado de veículos conectados e dispositivos médicos é apenas o início do problema. Produtos de software já mataram.

No início dos anos 80, a indústria de software pela primeira vez se mostrou letal. A máquina de diagnóstico médico Therac-25, aplicou uma overdose de radiação X em seis pacientes.

A tragédia com o Therac-25 teve muitas causas, entre elas:

- Reuso de software do modelo anterior, o Therac-20.
- Mudanças na interface de usuário que não foram aplicadas nos subsistemas de controle.
- Condições de corrida (race conditions).

Três dos seis pacientes que receberam a overdose faleceram em resultado da exposição a radiação.

Mas isto foi consequência da falta de testes? De acordo com Leveson, o fabricante executou testes extensivos, mas talvez não os testes necessários. Simuladores de hardware foram o principal mecanismo de testes do software. É tentador assumir que se o software não existe bugs, ele deve ser seguro. Contudo isto muitas vezes não é verdadeiro.

No meio do seu relatório, Leverson faz uma afirmação em relação ao código assembly PDP-11 que pode ser aplicável aos dias de hoje:

> *Os erros foram relativos a más práticas de engenharia de software em uma máquina que depende de software para uma operação segura. Além disto, o erro de codificação em particular não é importante se comparado ao design inseguro do dispositivo.*

O problema desta afirmação, quando aplicada aos design atual de software, é que normalmente um produto de software está além da nossa habilidade de compreendê-lo com precisão. Nós somos confrontados com um paradoxo. No desenvolvimento ágil, nós entregamos pequenos conjuntos de funcionalidades de forma a entender melhor os requisitos, design e execução do código. Mas como cada conjunto é adicionado aos conjuntos anteriores, as interações dos nossos sistemas compostos por sistemas crescem exponencialmente, diminuindo a nossa compreensão, entendimento e confiança do que entregamos.

Maturidade leva tempo

FTA, STAMP e modelos UML são ferramentas que nos ajudam a produzir sistemas mais confiáveis. Relatórios de falha na nossa indústria sugerem que devemos aumentar de forma agressiva a nossa preocupação com a segurança de nossos produtos, além de desenvolver a funcionalidade correta. Veículos wireless vulneráveis, dispositivos médicos expostos, redes de hospitais inseguras, babás eletrônicas vulneráveis, smart TVS que possibilitam que hackers escutem suas conversas são provas que devemos tornar o nosso mundo digital mais seguro.

LEIA ONLINE

LEIA ONLINE

A BATALHA POR SEGURANÇA NA INTERNET DAS COISAS

por **Eder Ignatowicz**

Quais são os desafios de segurança dos dispositivos de IoT? Qual o impacto que eventuais falhas na segurança destes dispositivos tem no nosso dia a dia? Quais são as diferenças entre a abordagem tradicional de segurança de um software e a segurança de um dispositivo de IoT? Estas são algumas das questões discutidas por Alasdair Allan em seu keynote no QCon Londres 2017 e resumidas neste artigo.

De acordo com Alasdair, o modo como enxergamos a privacidade será transformado na nova era da Internet of Things ("Internet das Coisas" ou IoT). Para muitos, ainda existe uma clara separação entre a Internet e o mundo real, contudo, gradualmente esta fronteira desaparecerá:

> *Num mundo onde tudo é smart (inteligente), em breve toda a sua vida será mensurada, calculada e avaliada. Logo, a Internet não se tratará apenas do seu e-mail ou das fotos do seu gato, mas sim da sua frequência cardíaca, sua taxa de respiração e como foi o seu sono na noite anterior.*

Aliadar avalia que a pressa para criar dispositivos de IoT e conectá-los na Internet teve como consequência uma baixa preocupação com a arquitetura destes dispositivos, principalmente em relação à privacidade e segurança. Ele enfatiza:

> *Devemos consertar a Internet das Coisas antes que ela se torne uma ameaça a própria Internet.*

Um dos principais erros no desenvolvimento de tecnologias de Internet of Things ("Internet das Coisas" ou IoT) é que o paradigma tradicional de segurança, que tem como premissa evitar o acesso físico a um dispositivo, não é aplicável a estes dispositivos pois um dos pontos chave desta tecnologia é que, quando necessário, o usuário possua acesso físico ao dispositivo.

Para ilustrar esta vulnerabilidade, durante a palestra foram apresentadas uma série de exemplos como:

- Como a indústria hoteleira expõe seu usuários em dispo-

sitivos IoT como radios, luzes inteligentes, portas, etc.;

- O ataque a privacidade dos Cloudpets (um urso de pelúcia IoT), onde foram expostos emails, senhas, imagens e milhões de minutos gravações de voz entre pais e filhos;
- A Stuxnet, worm que infectou silenciosamente uma série de computadores industriais e causou graves danos ao programa nuclear Iraniano.

Outra questão apontada por Aliadar é em relação a longevidade dos dispositivos de IoT. Segundo o cientista, estes dispositos são criados com o mindset do "Vale do Silício", baseado na ideia que estes equipamentos serão provavelmente substituídos em um ou dois anos. Contudo, o autor apresenta a seguinte reflexão:

> *Qual foi a última vez que você substituiu sua cafeteira? Qual foi a última vez que você substituiu todas a lâmpadas da sua casa? E todas as suas fechaduras?*

Alidar afirma que os dispositivos de IoT vão permanecer nas nossas vidas de 10 a 20 anos, pois esta é a expectativa de vida de carros, refrigeradores e fogões. Desta forma, a arquitetura de software e hardware destes dispositivos deve ser projetada para ser atualizada e mantida durante todo o ciclo de vida de um projeto. Mas o que acontecerá se estas empresas falirem antes deste tempo? Os dispositivos IoT irão parar de funcionar?

Podemos extrair dois pontos principais da mensagem deste keynote. O primeiro é que a segurança de dispositivos de IoT é responsabilidade dos desenvolvedores e esta deve ser uma preocupação desde o início do projeto dos dispositivos, pois estes são parte importante da nossa sociedade e não podemos deixá-los vulneráveis.

O segundo é que a arquitetura de software destes dispositivos afetará diretamente o modelo de negócio da empresa. Atualmente, o modelo escolhido por grande parte das empresas de IoT é simples: consumidores compram o dispositivo e recebem, sem nenhum tipo de assinatura mensal ou anual, todo o suporte dos serviços de cloud, pois estes são exatamente o que tornam este device inteligente.

Os fabricantes fornecem este serviço gratuitamente pois acreditam que os custos de manutenção destes serviços em nuvem serão baixos e que, novos consumidores ajudarão a cobrir estas despesas. Infelizmente, na prática, foi provado que esta expectativa é excessivamente otimista.

Alidar conclui afirmando que os desafios de IoT podem ser divididos em 3 áreas: segurança, ciclo de atualizações e standards (padrões) e atualmente, a comunidade de IoT demonstra preocupação somente com a consolidação de padrões de arquitetura, fato que surpreende o palestrante pois em sua opinião os outros dois desafios são, de longe, mas importantes.não são criados com os mesmos padrões, por exemplo, de um sistema bancário, ou do software que é produzido na Microsoft.

Enquanto buscamos umaInternet das coisas onipresente, quando somos desafiados com uma nova e perturbadora vulnerabilidade nos questionamos: "Por que alguém não antecipou isso?" Sergey Lozhkin, da Kaspersky Lab, relatou falhas de segurança perturbadoras em equipamentos médicos. Lozhkin concentrou sua atenção nas clínicas para determinar a vulnerabilidade desses equipamentos.

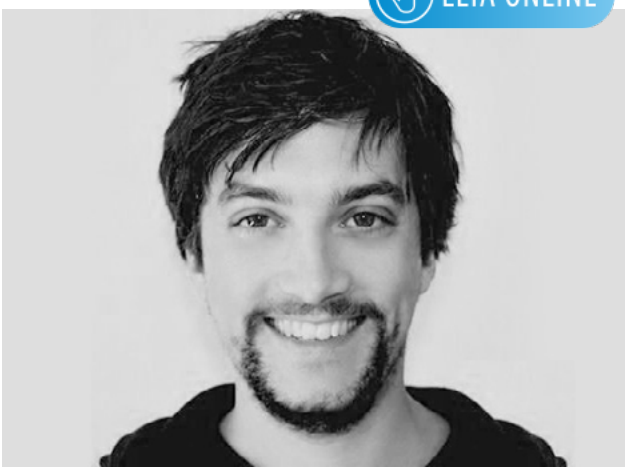
LEIA ONLINE

PHILIPP JOVANOVIC FALA SOBRE SEGURANÇA EM IOT, NORX E BLOCKCHAIN

por **Mathieu Bolla**, **Philipp Jovanovic**, traduzido por **Eder Ignatowicz**

DotSecurity é uma conferência de segurança para desenvolvedores "não especialistas em segurança". Alguns dos melhores hackers estavam presentes em Paris no dia 21 de abril de 2017, para falar sobre princípios de segurança, ferramentas e práticas que todo desenvolvedor deveria conhecer.

LEIA ONLINE



Nesta entrevista, originalmente publicada no InfoQ Francês, o especialista em criptografia Philipp Jovanovic - École polytechnique fédérale de Lausanne (EPFL), é entrevistado por Mathieu Bolla e fala sobre NORX, segurança IOT, Blockchain e como se manter seguro online.

InfoQ: Olá Philipp, meu nome é Mathieu Bolla, engenheiro de software na LesFurets.com e responsável pela segurança de dados de nossos clientes. Você é um especialista em criptografia na École polytechnique fédérale de Lausanne. Você poderia se apresentar em poucas palavras?

Olá Mathieu, prazer em conhecê-lo. Desde 2015, eu trabalho como pesquisador de pós-doutorado na EPFL's Decentralized e no laboratório Distributed Systems (DEDIS), onde colaboro com o Bryan Ford e seu time. Antes disso, me formei PhD em criptologia(cryptology) na University of Passau, Germany. Meus interesses de pesquisa incluem criptografia aplicada, segurança da informação e sistemas distribuídos e descentralizados.

InfoQ: Você recentemente trabalhou no NORX, um dos candidatos para o CAESAR (Competição de Criptografia Autenticada: segurança, aplicabilidade e robustez) que deve se encerrar no final de Dezembro de 2017. Eu entendo que o objetivo é simplificar a vida dos desenvolvedores provendo métodos de criptografia e autenticação em uma única solução, reduzindo os riscos do uso, por exemplo, de duas soluções contraditórias. Você pode nos falar mais sobre o NORX? Devemos utilizá-lo imediatamente? E em quais casos de uso?

O NORX é um novo algoritmo de encriptação autenticada (authenticated encryption) com suporte a associação de dados que tem como objetivo prover alto nível de segurança, boa performance tanto em software e hardware, e além disto, funcionalidades de segurança adicionais como resistência inerente a ataques timing side-channel (canal secundário). Nosso objetivo é manter o design geral do algoritmo o mais simples possível, buscando reduzir o risco de omitir alguma fraqueza de segurança, que, como você provavelmente deva imaginar, acontece muito facilmente. Embora nós estamos confiantes que o NORX é seguro, por enquanto, nós somente recomendamos seu uso em testes. Para usos reais, nós aconselhamos que aguarde até o final da competição CAESAR e, enquanto isto, utilize o AEAD ciphers (padrão atual) como AES-GCM ou ChaCha20-Poly1305.

InfoQ: Quando o vencedor do CAESAR for definido, você acredita que ele se tornará o padrão? Qual o impacto que os desenvolvedores podem esperar? Você acredita que ele substituirá por exemplo, o "HMAC over HTTPS messages" (HMAC sobre mensagens HTTP) que é utilizado pelos principais provedores de cloud, mas que são difíceis de se implementar de forma correta e muitas vezes inseguros?

Na verdade, o CAESAR não vai selecionar um vencedor entre os diversos algoritmos, mas sim recomendará um portfólio de ciphers categorizados entre sua aplicabilidade para certos cenários, como por exemplo software, hardware e dispositivos com recursos limitados. Uma vez que a competição se encerre, eu espero que nós tenhamos uma aplicação em larga escala do portfólio final do CAESAR, substituindo construções AEAD antigas como AES-CBC+HMAC e AES-GCM. É claro que esta mudança não acontecerá repentinamente, mas sim de forma gradual durante um longo período de tempo. Até onde eu sei, não existem planos de que

os vencedores do CAESAR se tornem automaticamente o padrão, como exemplo a competição NIST's SHA3. Contudo, eu espero que organizações como IETF's Crypto Forum Research Group (CFRG) avaliem a padronização dos vencedores do CAESAR de forma a simplificar ainda mais a adoção.

InfoQ: De forma mais genérica, em um mundo onde agências de inteligência são suspeitas de espionar diversos alvos, exércitos executam ataques virtuais e grandes corporações da internet enfrentam ataques de pequenos grupos e indivíduos, qual a sua opinião sobre o estado atual da segurança da informação? Você acredita que o nível de ameaça se intensificou? Ou estamos apenas nos tornando mais conscientes disso e de suas consequências?

Devido ao papel central que os sistemas de TI desempenham na nossa sociedade, acredito que não seja nenhuma surpresa que eles sejam cada vez mais atacados por hackers, criminosos ou outros governos, seja para o crime cibernético, a ciberespionagem ou simplesmente para o prejuízo moral. Acredito que pode-se esperar que essa tendência continue devido a avanços como as barreiras reduzidas no acesso de ferramentas de hacking e tutoriais na Internet, a lucratividade crescente do crime cibernético, governos em todo o mundo atualizando sua ofensiva virtual e sua capacidade de vigilância e ainda, o ritmo crescente no qual as novas tecnologias são desenvolvidas e lançadas, muitas vezes de forma prematura, no mercado.

Embora a situação pareça sombria, ainda existe esperança. Como exemplo, a algum tempo atrás a IKEA lançou sua plataforma de iluminação inteligente, Trådfri, que possui uma arquitetura de arquitetura de segurança bem aceitável. A primeira vista, é surpreendente o fato que de, entre todas as empresas, é a IKEA que aponta a importância do investimento de um bom design de segurança para produtos de IoT. Contudo, a decisão da IKEA é compreensível: ao não enxugar os custos do seu produto de IoT, a IKEA reduz o risco de que seus dispositivos sejam hackeados em larga escala, o que forçaria a empresa a realizar um caro recall do produto e ainda afetaria substancialmente a imagem da empresa. Graças a esta decisão, o mundo provavelmente é poupado de experimentar uma botnet no Trådfri, com um número imenso de lâmpadas, cenário onde a escala da botnet do Mirai seria apenas uma piada. Eu acredito que seria importante que mais empresas (especialmente as de IoT) se espelhassem na estratégia da IKEA neste mercado.

Esta pequena história ilustra outro ponto importante: é crucial investir na educação de designers e desenvolvedores em segurança pois novos produtos de TI devem sempre ser projetados com os requisitos de segurança em mente, uma vez que patches de segurança são raramente efetivos em um produto de IoT.

Eu acredito que devemos aumentar a conscientização da população (no nível do usuário final) para questões de segurança. Este é outro assunto importante que devemos continuar trabalhando.

InfoQ: Como indivíduo e desenvolvedor, quais ameaças você acredita que serão dominantes este ano? Nós ouvimos muito sobre ransomware, ataques de phishing e IoT zumbis mas estes ataques são, pelo menos a princípio, evitáveis. Como você se protege? E de quais ameaças?

As ameaças que você mencionou são, pelo menos em teoria, evitáveis. Mas infelizmente nós não vivemos em um mundo perfeito e portanto, devemos esperar que nos próximos anos, sejam cometidos erros de design, deployment e uso do sistemas de TI. Então não deve ser uma grande surpresa, se testemunharmos em 2017 diversos ataques virtuais provavelmente afetando a próxima geração de produtos de TI como, por exemplo, carros conectados, dispositivos médicos, sistemas de realidade aumentada e de realidade virtual.

Para proteção dos meus dados e dispositivos eu uso as boas práticas já amplamente conhecidas: instalar as atualizações de software assim que elas sejam disponibilizadas, criptografar todos os meus discos, utilizar passwords diferentes em cada serviço através de um gerenciador de passwords, ativar autenticação 2-factor em todos lugares possíveis (para segurança adicional use uma Yubikey) e possua múltiplos backups criptografados. Para mensagens em dispositivos móveis, eu uso exclusivamente aplicativos com suporte a criptografia end-to-end como o Signal/Wire/iMessage/Whatsapp. Eu também tento, sempre que possível, utilizar nos meus e-mails criptografia PGP/SMIME. E principalmente, fique alerta para os links que você clica em seus e-mails e mensagens, pois mais de 91% dos ataques virtuais começam desta forma.

InfoQ: Eu sei que você também trabalhou com blockchain, mais especificamente o ByzCoin - uma extensão do Bitcoin e outras blockchains. Você comparou o throughput máximo de transações entre o Paypal e a VISA, e teve excelentes resultados comparados com a implementação atual do Bitcoin - capaz de realizar aproximadamente 7 transações por segundo. Você acredita que sua implementação ganhará tração e tornará as cryptocurrencies (criptomoedas) mais estáveis e populares?

Seria incrível se uma criptocurrency (criptomoeda) como o Bitcoin adotasse o ByzCoin, pois teríamos não somente uma melhoria na performance, mas também na sua segurança. Por diversas vezes, questionamos a comunidade de Bitcoin se ela estaria interessada em colaborar com a

adoção do ByzCoin, mas, infelizmente, pelo menos agora eles não parecem estar muito interessados. Contudo, sendo razoável, a transição para o ByzCoin seria somente um primeiro passo na solução dos problemas que as criptomoedas enfrentam atualmente. Entretando, este passo inevitavelmente aumentaria a urgência na solução de outros desafios enfrentados por estas modas como, por exemplo, o ByzCoin poderia aumentar a taxa de transações de Bitcoin em até duas ordens de grandeza (para ~ 700 TPS), o que também aumentaria os drasticamente os requisitos de armazenamento. Consequentemente, somente nodes (nós) com uma excelente infraestrutura poderiam continuar mantendo blockchains, o que centralizaria os Bitcoins. Além disso, outro fator crítico é o consumo de energia dos mecanismos de mineração e este deve ser abordado pois é fator crucial na redemocratização as criptomoedas.

InfoQ: Você menciona no seu artigo que é possível estender outras blockchains. Se você analisar o Ethereum, eu acredito que seu trabalho possibilitará lidar com uma base maior de contratos inteligentes (smart contracts) em uma taxa de transações maior que a atual. Este é o único fator limitante para o que pode ser implementado em uma blockchain com o Ethereum? Você acredita que seu trabalho já definiu este limite ou ainda há espaço para outros avanços?

O ByzCoin certamente poderia ajudar o Ethereum da forma que você descreveu, mas ele também possui outras limitações no sentido de que ele só pode escalar até um certo ponto, pois seu desempenho poderá degradar se o grupo de consenso crescer além de um determinado tamanho.

InfoQ: Agora, imagine o futuro. Recentemente você trabalhou no NORX para a competição CAESAR e em seguida trabalho com blockchain. Quais são seus planos para o futuro? Você pode nos dizer quais projetos você planeja para o próximo ano? O que há de mais moderno nos laboratórios de criptografia?

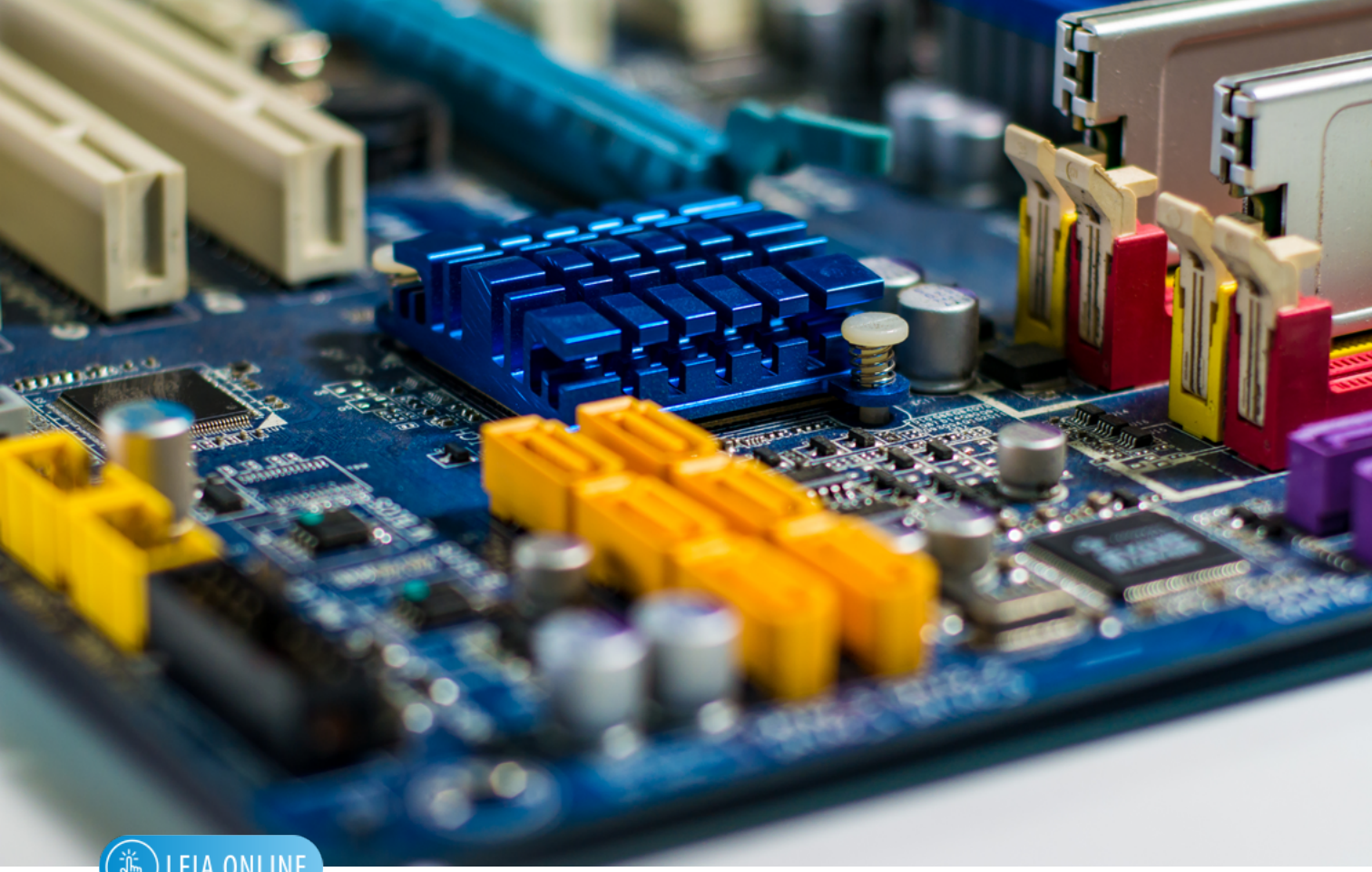
Claro! Existe uma série de projetos de pesquisa que eu estou atualmente envolvido.

Eu ainda trabalho com problemas de criptografia simétrica, como por exemplo nosso mais recente trabalho relacionado aos riscos potenciais no uso do AES-GCM no TLS. Outro projeto que tenho trabalhado é em construções Masked Even-Mansour (MEM) que pode ser utilizado para especificar modos de criptografia seguros e altamente eficientes e utilizam uma abordagem de design diferente que, por exemplo, o NORX.

Além disto, eu continuo minha pesquisa sobre segurança e escalabilidade em sistemas distribuídos no projeto cothority. Um dos subprojetos desta área, que apresentaremos a sua primeira fase numa palestra na conferência IEEE Security and Privacy 2017 é sobre a geração de aleatoriedade de distribuída de uma forma escalável, sem tendência (unbiasable) e verificável por terceiros. O projeto tem uma série de aplicações como por exemplo voto eletrônico, loterias e tecnologia blockchain. Nos próximos passos deste projeto, nós pretendemos explorar outras aplicações deste sistema e tornar o protótipo atual em um serviço online, possibilitando que toda a população tenha acesso a aleatoriedade (randomness) com as características citadas anteriormente.

Outro ponto que nós temos trabalhado no DEDIS são novas maneiras para gerenciamento seguro de identidades digitais, que são cruciais para o descoberta e deploy de chaves de criptografia (encryption), para um modelo mais seguro de procedimentos de atualização de software e encontrar melhores alternativas proof-of-work e proof-of-stake para mecanismos contra fraude (anti-sockpuppet) que são a base de todas as criptomoedas modernas.

LEIA ONLINE



LEIA ONLINE

IOT E O TERCEIRO CONSUMIDOR: CRIANDO SERVIÇOS PARA DISPOSITIVOS LIMITADOS

por **Wellington Mariusso**

Aplicações Web modernas trabalham num modelo em que a API tem um papel central. De uma API bem definida e padronizada é possível criar uma camada de apresentação Web num modelo de Single Page Application, uma experiência mobile com Apps ou a integração com sistemas de terceiros.

Num primeiro impulso, pode parecer que usar uma API pré-existente também seria a solução ideal para conectar dispositivos físicos a uma lógica de negócio pré-existente - criando uma experiência real de Internet das Coisas, como permitir que um cliente compre produtos usando um botão físico.

É preciso porém avaliar com mais cuidado se esse é o melhor caminho. IoT tem suas particularidades que precisam ser atendidas.

Dispositivos de IoT em geral são devices de custo razoavelmente baixo, sem tela, muitas vezes com conectividade limitada e distribuídos numa escala maior que aplicações mobile ou web. As projeções indicam que num futuro não distante a quantidade de dispositivos desse tipo deve superar a quantidade de smartphones em proporção de 4:1. Vale a pena considerar os impactos desse cenário.

Há muitos tipos de dispositivo, mas para poder exemplificar as limitações, vamos dividir aqui em dois grupos de estudo: dispositivos semelhantes a Single Board Computers e Microcontroladores. Cada grupo tem suas vantagens e seus desafios.

Na linha de Single Board Computers, o representante mais bem sucedido no momento é o Raspberry Pi e suas variações. Um Raspberry Pi 3 tem muitas opções de conectividade (WiFi, Bluetooth, Ethernet) e capacidade de processamento invejável. É possível trabalhar em cima de um Sistema Operacional como Linux ou Windows e a experiência de se trabalhar com um dispositivo desse tipo é muito parecida com a de se trabalhar com um pequeno servidor virtual na nuvem.

As ferramentas são similares às que usamos no dia-a-dia do desenvolvimento de aplicações convencionais, tendo muitas opções de escolha. Linguagens de alto nível como Java, Javascript, Python e Ruby estão disponíveis e funcionam exatamente da mesma maneira que em um ambiente na nuvem. Os mesmos frameworks e bibliotecas a que temos acesso no momento de construir ou consumir micro serviços também estão disponíveis em um Single Board Computer. O desenvolvimento é fácil e a compatibilidade tecnológica é alta.

Dadas tais características, é possível usar uma API pré-existente de maneira muito natural. Não há grandes diferenças entre um Single Board Computer e um Smartphone ou uma aplicação Web.

Por outro lado, há características que talvez não sejam as mais desejáveis para uma solução de IoT. O custo pode ser elevado (um Pi3 é vendido por aproximadamente 35 dólares) e o consumo de energia elevado torna o uso de baterias improvável.

Do outro lado, temos um outro grupo de dispositivos em soluções de IoT que é o grupo de Microcontroladores. Nessa linha, o representante mais bem sucedido para protótipos e iniciantes é o Arduino.

Soluções baseadas em microcontroladores têm vantagens como consumo de energia muito reduzido e tempo de inicialização extremamente rápido. É possível combinar essas duas características de maneira formidável através da aplicação de duty cycles: ciclos em que o microcontrolador realiza uma determinada tarefa, entra em sono profundo com um consumo de energia extremamente reduzido (~1000x menos que um Single Board Computer) e acorda novamente depois de um período para retomar suas atividades. Combine isso a microcontroladores otimizados para baixo consumo e é possível construir soluções que podem operar por anos em uma única carga de bateria e ocupando pouco espaço. O custo também tende a ser bastante acessível. É possível encontrar dispositivos desse tipo por custos próximos a 2 dólares. Perfeito para sensores em locais de difícil acesso ou ambientes em que não há energia elétrica constante.

As limitações são muitas, entretanto. Tais dispositivos, às vezes chamados de Constrained Devices, estão bastante distantes da realidade do desenvolvedor de aplicações para a nuvem. Não é incomum trabalhar com apenas 2K de memória RAM. O desenvolvimento é com com linguagens de mais baixo nível (C, C++, Sketch, Assembly) e normalmente usando as bibliotecas e ferramentas disponibilizadas pelo próprio fabricante do microcontrolador. Há pouca portabilidade: código escrito para um modelo de dispositivo dificilmente pode ser reutilizado em outro modelo. Há ainda as dificuldades relacionadas a conectividade: frequentemente tais dispositivos estão conectados em rede com baixas taxas de transferência e latência alta. Uma sessão de depuração remota é inviável.

Dadas tais limitações, quando um desenvolvedor de firmwares para microcontroladores precisa interagir com uma API REST, muitas vezes opta por implementar parte do protocolo manualmente, para poupar recursos e evitar importar bibliotecas adicionais. Essas implementações podem ser um problema, uma vez que tratam apenas do mínimo necessário para fazer com que a solução funciona - sem tratar dos pormenores do protocolo. Se no futuro a API evolui ou migra para uma infraestrutura distinta, não há garantias de que as soluções que foram implementadas manualmente - ou que dependem de uma biblioteca pouco usada - serão capazes de lidar com a nova situação.

Um outro ponto de atenção está relacionado com o quão extensos são os cabeçalhos do protocolo. Uma chamada a uma API REST, especialmente se usada para enviar pequenos trechos de dados, gasta mais banda de rede transportando cabeçalhos e metadados do que informação per-se. E numa solução desse tipo, consumo maior de rede em geral implica em consumo maior de bateria - porque o microcontrolador e o rádio precisam ficar mais tempo ligado do que necessário.

APIs REST não são muito naturais do ponto de vista de um desenvolvedor de firmware para dispositivos com capacidade limitada e não são a melhor solução para interagir com tais dispositivos.

Se realmente é necessário exportar um endpoint REST para um dispositivo restrito, algumas práticas podem ser adotadas para minimizar as dificuldades envolvidas. Uma delas é considerar endpoints distintos para a aplicações e dispositivos. Isso permite, entre outras coisas, que a infraestrutura possa ser tunada para os clientes específicos. Dispositivos de IoT em geral precisam de timeouts mais generosos que browsers, por exemplo. Também é possível remover cabeçalhos não essenciais e eventualmente trafegar dados em formatos diferentes. Não é bom contar com a capacidade do dispositivo em tratar de compressão de dados: o melhor é trafegar o mínimo possível de informação, num formato bastante compacto.

Há alternativas, porém. Mecanismos que são capazes de cobrir o gap entre o que é natural para um desenvolvedor de aplicações na nuvem e o que é mais natural para um desenvolvedor de firmware mais limitado.

Uma alternativa é o uso de protocolos mais adequados, como é o caso de MQTT. O MQTT é um protocolo feito especificamente para troca de informações como nos cenários de dispositivos limitados. É um modelo assíncrono (pub/sub) com cabeçalhos bem mais enxutos que HTTP.

Além disso, por ser assíncrono, não exige uma resposta do servidor a cada mensagem enviada. Tráfego de informação em formato binário, muito mais compacto e natural do que JSON, é bastante simples em MQTT. Há um grande número de bibliotecas disponíveis para diferentes dispositivos e arquiteturas e a natureza do protocolo binário desencoraja que o desenvolvedor faça a implementação manualmente do lado do device. É frequentemente uma opção mais econômica em termos de recursos e mais robusta para soluções IoT com limitações de rede e bateria.

O ponto negativo dessa abordagem é ter que criar e manter um componente de software e de infra para realizar a interface entre o broker MQTT e a API já pré-existente.

Um outro caminho é o uso de Plataformas e Gateways de IoT. Tais plataformas fazem o trabalho de cobrir o gap entre o mundo cloud e o mundo do device de uma maneira que seja natural para ambos os lados, com pouco ou nenhum esforço de codificação envolvido. Também procuram adicionar serviços, como gestão de dispositivos, monitoração e atualização de firmware, entre outros aspectos de aceleração de construção de dispositivos. Aqui também há pontos a serem considerados. Há soluções que variam desde frameworks a modelos do tipo plataforma como serviço (PaaS), passando por gateways físicos que fazem esse papel. Cada uma dessas soluções tem seus desafios próprios. De uma maneira geral, é importante considerar especialmente o possível problema de vendor lock in. Qualquer solução que se apresente nesse cenário é superior quando trabalha com protocolos abertos e, preferencialmente, seja Open Source.

Em resumo, é possível pensar em disponibilizar uma API convencional para dispositivos em um cenário de IoT - mas deve-se fazê-lo levando em consideração as limitações dos dispositivos. Para alguns cenários, não há grandes preocupações: Single Board Computers tem o mesmo ferramental que servidores na nuvem. Para dispositivos extremamente limitados, como microcontroladores, considere uma abordagem distinta. Uma troca de protocolo ou o uso de uma plataforma para cobrir os gaps entre a API e o dispositivo podem prevenir muitos problemas e produzir melhores resultados.

LEIA ONLINE

LEIA ONLINE

DESAFIOS NO DESENVOLVIMENTO DE APIS E IOT NO MUNDO PROGRAMÁVEL

por Eder Ignatowicz

O mundo está gradualmente se tornando mais programável e a gama de soluções técnicas que temos à nossa disposição para criar essa mudança cresce a uma velocidade vertiginosa. O potencial do mundo programável , a empresa programável e até mesmo o humano programável é espetacular. Contudo, talvez precisemos voltar um passo atrás, e refletir sobre o que o comportamento da indústria.

Este é o tema da palestra que Steven Willmott, diretor sênior e head de infraestrutura de API Red Hat e ex-CEO da 3scale Inc, apresentou no último APIdays na Austrália e é resumida neste artigo.

Dentre os temas abordados, é discutido como APIs e os recentes avanços tecnológicos estão contribuindo para o mundo programável. É realizada uma reflexão sobre quais são os nossos objetivos com este avanço e por fim, são compartilhadas quais são as melhores práticas para sucesso nesta área.

Steven abre a palestra afirmando que estamos vivendo uma época sem precedentes, onde é possível construir praticamente qualquer ideia de negócio de forma simples, rápida e com pouco recursos. Ele afirma que toda a infraestrutura das empresas devem se basear em três pilares: integração, containers e APIs.

As APIs permitem que você crie ilhas de estabilidade em torno da sua empresa, para que você possa construir seu negócio sobre cada componente do seu sistema. E que containers se tornaram a maneira padrão de escalar o deploy da sua aplicação.

Integração entre diversos sistemas sempre foi tema central do mundo de TI, mas esta área está sofrendo uma importante mudança de paradigma. Do antigo cenário onde se integrava toda a arquitetura em um único ponto central, a integração evoluiu e vem sendo realizada de forma distribuída, ou seja, em todos os pontos da sua arquitetura.

Em seguida, Willmott demonstra como a tecnologia que

nós utilizamos diariamente, não existiam há 10 anos atrás. Como exemplo, o AirBnb que transformou o modo que pensamos em hospedagem, tecnologias eletrônicas de fechadura, que mudaram a forma que acessamos nossos lares e tecnologias como Amazon Echo e Alexa que mudaram a forma que nos comunicamos com a nossa casa.

Trazendo estas inovações para perspectiva de software, Steven aponta exemplos como a Stripe, que possibilita que uma aplicação aceite pagamentos com apenas duas linhas de código e ainda analise as métricas de negócio da empresa, tornando simples criar um negócio com recursos limitados, e a Fitbit, plataforma de tracking de dados de saúde que recebe mais de 400 bilhões de chamadas por mês em sua API e que possibilita ao usuário criar dashboards de monitoramento da sua saúde e atividade física, e ainda compartilhar estes dados de saúde com hospitais e clínicas.

Na perspectiva de hardware, ele afirma que dispositivos com tamanho reduzido, gps de baixo custo, RFIDs minúsculos e tecnologias como hololens da Microsoft revolucionarão a forma como vivenciamos o mundo. Como exemplo, ele afirma que estes dispositivos transformarão a vida de portadores de deficiência visual ou auditiva.

Contudo, toda esta funcionalidade torna a disponibilidade de nossos dispositivos e dados um fator crítico ao usuários, pois cada vez mais a população torna-se dependente destas tecnologias. E Steven faz a seguinte pergunta:

E o que estamos fazendo com toda esta responsabilidade e dependência das nossas tecnologias?

Ele afirma que são quatro áreas principais que devemos nos preocupar em relação ao nosso software e hardware: segurança, comportamento inesperado, deficiência tecnológica e impacto social.

São apresentadas alguns exemplos de brechas de segurança preocupantes, por exemplo, como um passageiro invadiu o sistema de entretenimento de um avião da United e enviou comandos para os motores da aeronave, como pesquisadores conseguiram desligar o motor, acionar o volante e sistemas de freios de um Jeep e do recente ataque do dyndns, realizado através da invasão de 10 milhões de dispositivos como câmeras e outros dispositivos de segurança.

O palestrante ainda aponta que nós, como engenheiros, devemos nos preocupar com o impacto que as nossas tecnologias terão na sociedade, principalmente em relação a qualidade de vida da população. Como exemplo, ele apresenta uma imagem de uma cafeteria totalmente automatizada, tornando todo o processo desumanizado. Ele faz uma importante reflexão da diferença entre tecnologias que automatizam 100% da produção, de tecnologias que tornam a população mais produtiva.

Nós estamos criando tecnologia que levam as pessoas a serem consumidores ao invés de usuários e geradores de valor. Isto terá um impacto direto na economia e talvez este processo nos desumanize, pois perderemos as interações humanas.

Ele ainda afirma que o sentimento geral da comunidade de software é que pouco pode ser feito em relação a este assunto. Contudo, segundo Steven, nós devemos parar e refletir nos porquês desta realidade.

Para entender os porquês desta realidade, ele apresenta o conceito da TED talk de Simon Sinek "How great leaders inspire action" (Como grandes líderes inspiram ação), que segundo Steven, entender o porquê das ações que você realiza é o que causa impacto na forma que você lidera um time e até nas suas atitudes.

Ele afirma que a grande maioria das pessoas entende o que está sendo feito, provavelmente sabe como está fazendo, mas possui apenas uma vaga ideia do porquê das iniciativas.

Isto é ainda mais claro no mundo corporativo, pois grande parte das empresas apenas se preocupa com o que está sendo desenvolvido (um dispositivo com uma API inteligente) e fala sobre como está fazendo (alta escalabilidade, UX, API), mas normalmente esquece de comunicar o porquê do desenvolvimento.

E Steven afirma que isto é um risco, pois ele acredita que todo criador de um produto ou tecnologia no fundo sabe o porquê que criou a sua empresa, mas falha na comunicação ao cliente e sua equipe. Ele afirma que entender e comunicar claramente o porquê por trás da sua tecnologia ou produto é um fator crucial ao sucesso da sua iniciativa.

E isto ainda é mais claro em iniciativas de IoT, "programmable world" e smart devices (dispositivos inteligentes). Ele afirma que raramente explicamos o porquê de nossas iniciativas para a população, que ainda assim consome nossos produtos sem saber ao certo a razão da existência destas tecnologias.

Os riscos desta falha de comunicação fica claro quando temos que justificar ao grande público eventuais falhas nestes dispositivos, como exemplo quando ocorre um acidente em um carro autônomo. Será que se a maioria da população conhecesse os porquês do desenvolvimento destas tecnologias, não facilitaria a aceitação de eventualidades?

Steven, segue tentando entender o porquê queremos tornar o mundo mais programável. Ele elenca quatro justificativas:

- Melhorar o futuro da humanidade;
- Melhorar a vida humana;
- Melhoria sócio-econômica;

- Criar pequenas melhorias diárias;

E ele convida o público (e o leitor), a pensar no seu próprio porquê.

Uma vez definido o seu porquê (Why), Steven afirma que é o momento de definir o seu "o que"(What) e o seu "como" (How). Ele fala que não pode ajudar o público com o "o que", pois depende do domínio de trabalho de cada um, mas pode compartilhar o que ele acredita serem os princípios para o "How" (como).

O princípio para o "como" devemos desenvolver nossos produtos, deve inevitavelmente passar por algum tipo de código ético. Fundamentalmente, todas as profissões que afetam a vida das pessoas possuem um código de ética, como exemplo o direito, medicina e educação possuem um código de ética rigoroso e caso um profissional viole este código, ele é excluído da profissão.

Mas isto não ocorre para o desenvolvimento de software, o que é até positivo em alguns cenários. Contudo, Steven acredita que os profissionais devem se "auto regular", pois as tecnologias cada vez mais fazem parte da vida das pessoas e as afeta diretamente e profundamente.

Além disto, é apresentado quatro pilares para o mundo programável Continuous Improvement (melhoria contínua), Graceful Degradation (capacidade de um sistema de se manter funcionando com funcionalidades reduzidas mesmo em situações adversas), Radical Distribution (distribuição em larga escala) e Componentes (não somente soluções).

Continuous Improvement é muito importante no mundo das APIs. Suas APIs não devem quebrar de uma versão para outra do seu sistema. Se isto for necessário, deve se realizar todo um planejando e utilizar alguma estratégia, por exemplo versionamento. Ele ilustra este conceito falando sobre a API do Stribe, onde cada cliente é atrelado a uma versão específica da plataforma e só faz o upgrade se o cliente decidir.

Em relação ao Graceful Degradation, Steven afirma que APIs ainda possuem um contrato rígido. Como exemplo, se sua API consome quatro serviços de backend e somente um deles falha, devemos retornar um erro 500 ou os dados parciais para o cliente? Segundo Steven, devemos retornar dados parciais de forma sistemática e como exemplo, ele cita o exemplo do Netflix e o chaos monkey.

Steven também afirma que devemos distribuir nossos sistemas em microservices e devemos ter em mente a distribuição de times, data centers e qualquer tipo de recurso essencial aos nossos sistemas. Neste cenário o projeto de uma boa API é fundamental para o sucesso deste modelo arquitetural.

O quarto pilar para o mundo programável é que devemos construir componentes e não somente soluções fechadas. Embora seja natural disponibilizarmos software empacotado para uma melhor experiência para nossos clientes finais, no mundo de IoT isto impossibilita que este usuário crie as suas próprias soluções através da composição de componentes.

Entregar ao usuário final uma solução fechada e pronta, exclui o cliente do desenvolvimento da solução e ignora as particularidades de cada cliente, aplicando uma solução genérica a um problema específico.

Segundo Steven, APIs e IoT devem se preocupar em serem o mais abertas e componentizáveis possíveis, tornando simples a conexão com outros dispositivos, se adaptando a realidade do cliente e aumentando o valor da solução final. O desenvolvedor de um dispositivo de IoT deve se preocupar em participar da solução final de seu cliente e não ver seu cliente como consumidor de uma solução fechada.

Steven conclui afirmando que é importante que cada desenvolvedor de IoT e APIs tenha coragem e tenha orgulho das soluções que desenvolve. Ele afirma que estes desenvolvedores são pioneiros nesta área e devem tomar para si a responsabilidade do software que constroem.

LEIA ONLINE

eMag