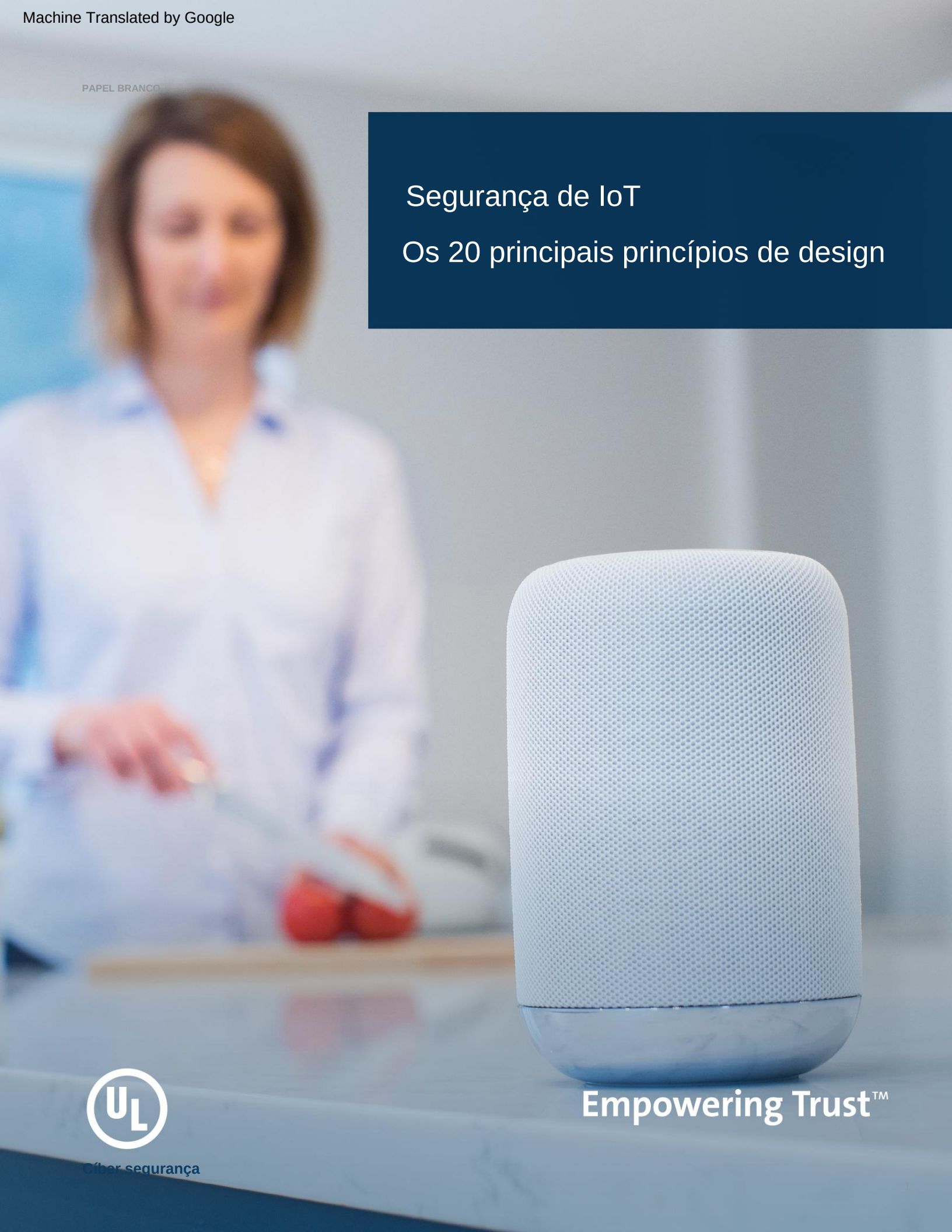
Segurança de IoT

Os 20 principais princípios de design



Empowering Trust™

Cíber segurança

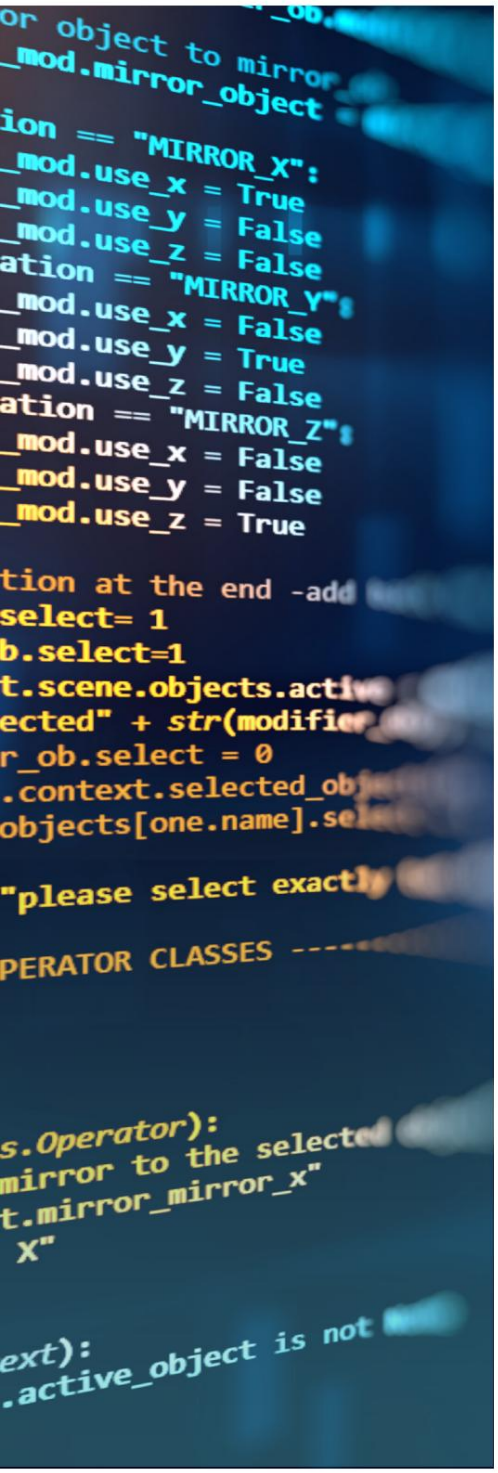
Executivo Resumo

Num mercado competitivo, é importante encontrar um diferenciador para os seus produtos e, muitas vezes, procura-se uma vantagem competitiva através da adição de funcionalidades inteligentes e da ligação de produtos a redes empresariais ou à Internet. No entanto, à medida que novas funcionalidades e ligações são adicionadas, a segurança de tais sistemas é frequentemente degradada.

Além disso, a segurança dos produtos conectados está se tornando cada vez mais uma questão de interesse organizacional e até nacional. Malware que pode assumir o controle e subverter as operações de sistemas conectados tem sido usado para lançar alguns dos maiores ataques já vistos na Internet. A natureza conectada desses sistemas também significa que a segurança deve ser considerada para quaisquer aplicativos executados em sistemas separados, como serviços em nuvem e telefones de consumo.

É claro que pode ser difícil adequar a segurança a requisitos de tempo e orçamento cada vez mais apertados para o desenvolvimento de produtos. Felizmente, existem algumas etapas simples que podem ser seguidas para aumentar a segurança dos sistemas conectados. Essas etapas descritas são organizadas primeiro com os requisitos mais importantes e é recomendado que sejam abordados como prioridade inicial para todos os aspectos de um sistema – produto, sistema, nuvem e aplicativo.

Cinco principais prioridades



1 Fornece uma substituição manual para quaisquer operações críticas de segurança

Os recursos avançados são ótimos quando funcionam. No entanto, às vezes esses recursos falham – muitas vezes sem culpa do próprio sistema. A rede local do cliente pode estar mal configurada ou a conexão com a Internet pode ser interrompida. Nesses casos, é importante que qualquer falta de funcionalidade que isto cause não resulte num problema de segurança para o utilizador final. Os exemplos podem ser o fornecimento de um backup de chave física para uma fechadura inteligente ou um recurso de cancelamento manual e limitação de segurança em um termostato IoT.

2 Garanta que os parâmetros que possam comprometer o sistema (chaves criptográficas secretas ou privadas, senhas, etc.) sejam exclusivos por dispositivo

As senhas usadas para recursos sensíveis à segurança devem ser exclusivas por dispositivo. Uma senha que todo mundo conhece não é exatamente uma senha. Existem soluções simples para isso, no entanto. Por exemplo, senhas seguras podem ser geradas aleatoriamente e impressas no adesivo com o número de série do dispositivo. Se o dispositivo não for facilmente acessível durante a operação normal, considere fornecer esse adesivo dentro do manual ou guia de início rápido, que pode ser retirado e colocado em algum lugar que o usuário não esqueça. É claro que sempre haverá cenários em que os clientes esquecerão ou perderão senhas; portanto, métodos de recuperação do sistema, como um botão de redefinição física que entra no modo de recuperação de senha quando pressionado, também devem ser implementados com segurança.

Quaisquer chaves criptográficas secretas (simétricas) ou privadas (assimétricas) também devem ser gerenciadas como exclusivas para cada dispositivo ou aplicação. Existem algumas maneiras pelas quais as chaves criptográficas podem ser determinadas, mesmo quando aparentemente estão sendo armazenadas e gerenciadas dentro de um dispositivo seguro. Freqüentemente, esses métodos não valem a pena para extrair chaves de um único dispositivo, mas se a mesma chave for usada em milhares de dispositivos, a economia de tal ataque muda consideravelmente.

3 Teste o sistema para ter certeza de que ele está livre de vulnerabilidades conhecidas e exploráveis antes do lançamento

O software em dispositivos conectados, aplicativos e serviços em nuvem é frequentemente composto por vários componentes de software, incluindo software existente (como código-fonte aberto e bibliotecas de terceiros), bem como protocolos e funções comumente usados (como bancos de dados). Cada um desses componentes de software pode ter suas próprias vulnerabilidades e é importante que seja feita uma verificação de vulnerabilidades conhecidas antes de lançar qualquer sistema.

Isto pode ser conseguido através de vários utilitários de software e serviços de digitalização para sistemas baseados em nuvem. Na indústria de cartões de pagamento (PCI), por exemplo, procure fornecedores que tenham passado nos requisitos do ASV, que fez um bom trabalho ao fornecer uma validação mínima de fornecedores de digitalização.

4 Permita atualizações de software e garanta que elas sejam autenticadas criptograficamente antes da instalação e execução. Implemente recursos anti-reversão para evitar a instalação de versões anteriores vulneráveis de firmware

Não importa quão bem o software seja projetado ou testado, sempre haverá bugs e vulnerabilidades que serão perdidas ou descobertas após o produto ser enviado. É importante permitir a atualização do software para garantir que ele possa ser corrigido quando algum desses bugs for encontrado. No entanto, se isto não for implementado com cuidado, pode levar a vulnerabilidades adicionais, onde um malfeitor pode instalar o seu próprio software no dispositivo para impedir o seu funcionamento normal.

Para evitar isso, as atualizações de software devem ser autenticadas criptograficamente. Isso geralmente é conseguido através do uso de uma assinatura digital na imagem do firmware do sistema, que pode ser verificada pelo firmware original (ou carregador de inicialização do dispositivo) antes da instalação. O uso de uma assinatura digital baseada em um algoritmo de chave pública (como RSA ou DSA) garante que os próprios dispositivos não precisem da parte da chave (privada ou secreta) usada para gerar os dados de autenticação.

Se, em vez disso, for usado um sistema de chave simétrica, como um (H)MAC, essa chave secreta será necessária em cada dispositivo. Isso significa que o acesso ao firmware em um dispositivo oferece a capacidade de criar assinaturas de firmware válidas para qualquer dispositivo (a menos que haja uma chave exclusiva por dispositivo, o que não é viável para sistemas IoT). Portanto, a criptografia de chave pública é fortemente recomendada.

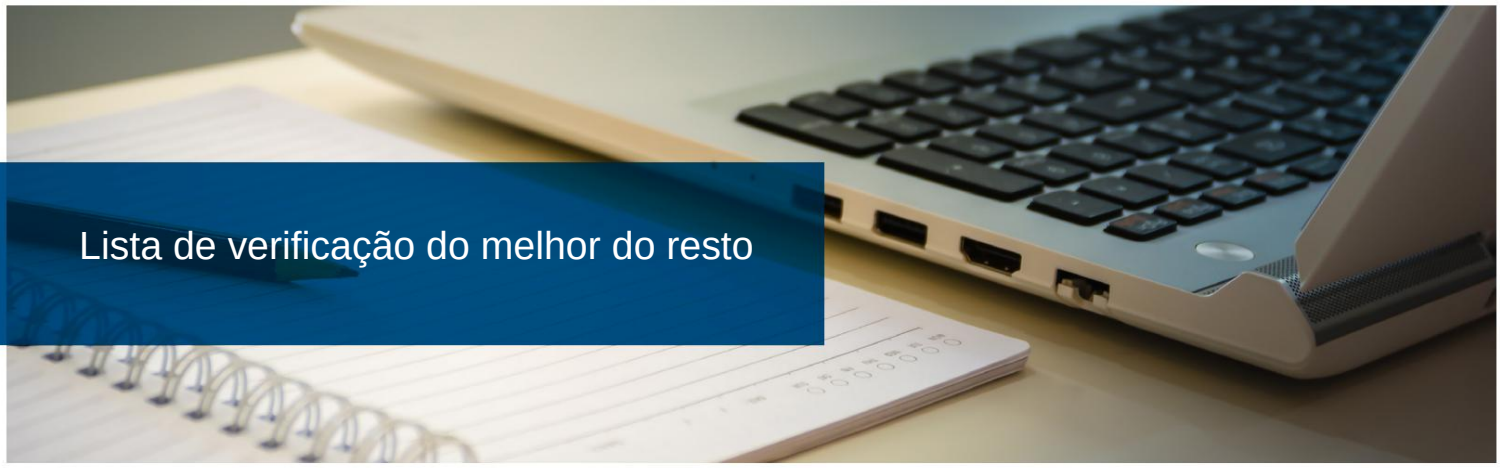
Também é importante incluir métodos para evitar que um malfeitor instale uma versão anterior do firmware, o que restabeleceria quaisquer vulnerabilidades corrigidas. Isso pode ser feito incluindo um número crescente (monotônico) em cada versão de firmware verificada antes da instalação para garantir que a versão do firmware que está sendo tentada a ser instalada não seja mais antiga que a versão atual no dispositivo.

5 Use protocolos de segurança padrão do setor com padrões de práticas recomendadas para qualquer conexão remota ou sem fio e autenticação de conexões para serviços de gerenciame

É vital proteger as comunicações que possam estar sujeitas a intercepção ou modificação utilizando um protocolo de segurança padrão da indústria, como TLS ou WPA2. Além disso, o uso exato do protocolo de segurança também é importante – por exemplo, é possível usar o TLS e ainda ficar inseguro se ele não estiver configurado corretamente. Esses protocolos permitem a autenticação de conexões, mas somente quando implementados corretamente. Portanto, a validação de certificados ou a fixação de certificados deve ser usada para garantir que a conexão seja segura e privada.

Use a versão mais recente do protocolo e da biblioteca de software e monitore quaisquer alterações para que patches possam ser fornecidos quando os problemas forem corrigidos. A utilização do protocolo de segurança deve abranger todas as comunicações sempre que possível, independentemente de estarem ou não relacionadas com a segurança. Isto tornará o comprometimento do sistema mais difícil, pois qualquer malfeitor deve primeiro comprometer o protocolo de segurança antes de obter acesso para tentar comprometer o dispositivo.

Este requisito abrange também a utilização de protocolos sem fios, onde a utilização de protocolos de segurança como o WPA2 é igualmente importante. Pode ser necessário permitir que os clientes desativem os recursos de segurança, mas considere fornecer-lhes orientações sobre por que isso não é recomendado.



Lista de verificação do melhor do resto

6 ão armazene senhas em texto não criptografado

É do conhecimento geral que as pessoas tendem a reutilizar palavras-passe, pelo que uma palavra-passe extraída de um único sistema comprometido pode ser utilizável noutros sistemas, contas e serviços online. Em vez de texto não criptografado, as senhas devem ser armazenadas usando um algoritmo “unidirecional” e computacionalmente intensivo, como o BCrypt.

Para qualquer ambiente de nuvem, este item deve ser considerado parte dos cinco principais requisitos.

7 Autentique interfaces de acesso remoto e gerenciamento de sistema com limites de sessão e tempo limite

O acesso a um sistema, seja a um dispositivo fora da rede local ou a um sistema em nuvem, deve ser autenticado para evitar o acesso de partes não autorizadas. Para sistemas back-end ou baseados em nuvem, considere implementar medidas de segurança de dois fatores, como aquelas fornecidas por tokens e software compatíveis com FIDO. Senhas de uso único baseadas em SMS podem ser implementadas, mas estão sob ataque crescente e métodos mais novos e mais seguros são recomendados.

Para conexão em redes locais a partir de um sistema externo, considere conexões VPN ou roteamento de dados através de um túnel de conexão TLS que possa fornecer autenticação (e onde a validação/fixação de certificado seja executada conforme necessário).
As conexões locais podem exigir apenas uma senha, mas também podem fornecer autenticação por proximidade física, como uma conexão Bluetooth/NFC ou por meio de um botão físico que deve ser pressionado para acessar o serviço. Se for usada uma rede sem fio localizada, certifique-se de que as práticas recomendadas de segurança sejam seguidas.

Interfaces de depuração, como JTAG e emulação no circuito, devem sempre ser desabilitadas em dispositivos de produção. Embora o acesso a essas interfaces exija acesso físico local, habilitá-las simplifica muito o trabalho de um malfeitor no desenvolvimento de ataques.

Esses serviços de gerenciamento podem ter diversas funções, desde virar uma câmera para apontar em uma direção diferente até carregar novos certificados, firmware ou outros recursos relacionados à segurança. Se um dispositivo tiver vários recursos que podem ser acessados por meio de recursos de administração remota, considere fornecer diferentes níveis de autenticação para que seja possível isolar os recursos relacionados à segurança dos recursos do usuário.

Além disso, forneça um limite absoluto de tempo durante o qual quaisquer recursos de administração podem ser acessados durante uma sessão. Isso evita que as pessoas esqueçam que deixaram esses recursos ativados.

8 arantir que as metodologias de chave criptográfica gerem aleatoriedade suficiente

Gerar bons números aleatórios é realmente muito difícil, e o uso de números aleatórios ruins tem sido a fonte de muitas vulnerabilidades. A causa raiz geralmente se baseia em duas questões: primeiro, os sistemas de computação são determinísticos, o que significa que o mesmo programa, com as mesmas entradas, produzirá sempre a mesma saída; segundo, nós, como seres humanos, não somos muito bons em detectar a falta de aleatoriedade.

Não se pode confiar apenas na entrada de um dispositivo incorporado para produzir bons números aleatórios. Muitas vezes é melhor obter informações de diversas fontes. Uma delas pode ser uma função aleatória padrão, mas outras fontes também incluem os bits menos significativos de uma entrada A/D, tempo de tráfego de rede, tempo de busca no disco rígido, dados em milissegundos de uma fonte de relógio em tempo real, tempo de entrada do usuário, etc. Eles podem então ser combinados e fornecidos como uma semente para um gerador de números pseudo-aleatórios, como aqueles descritos no NIST SP 800-90A.

As chaves criptográficas não devem ser geradas diretamente a partir de senhas. É melhor usar a senha para permitir o acesso ao uso de uma chave gerada a partir de um processo forte de números aleatórios.

9 Detalhe todos os dados do cliente – incluindo áudio, vídeo e detalhes pessoais – que podem ser exportados para sistemas em nuvem ou terceiros. Forneça uma opção para tal coleta

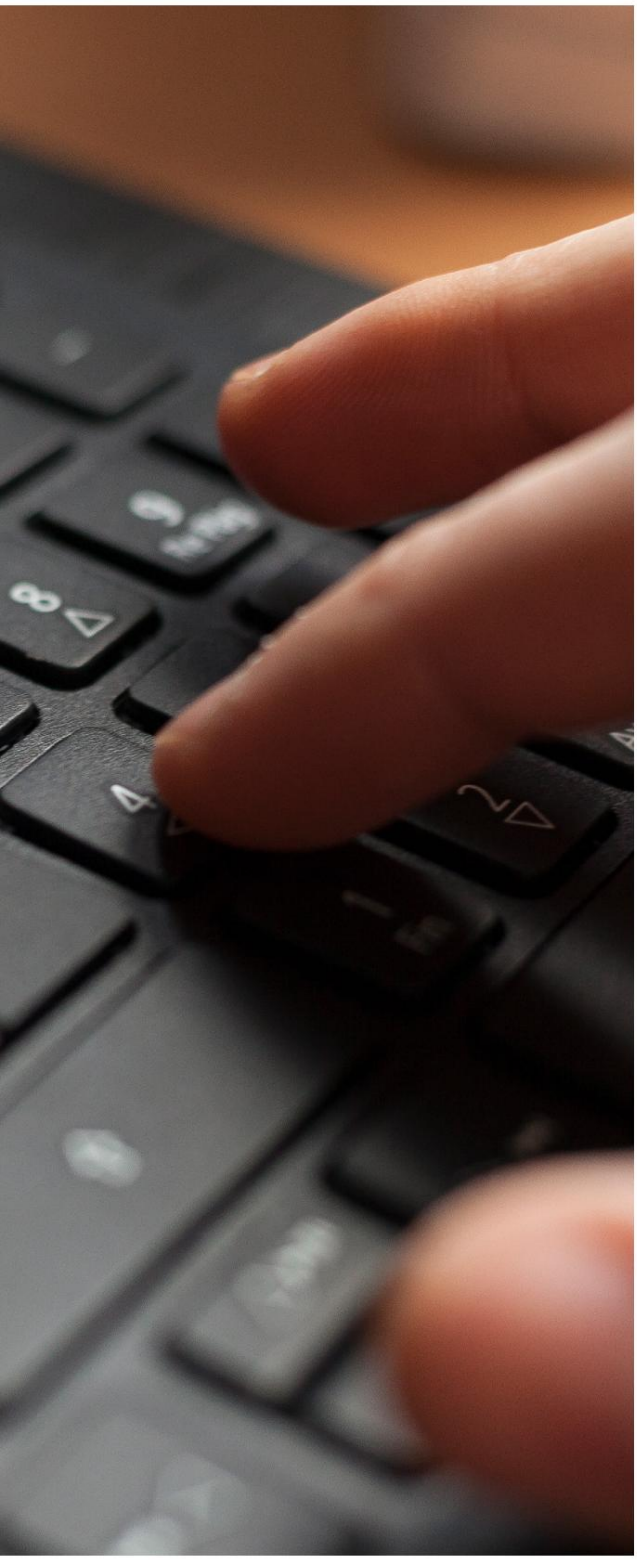
Muitos sistemas fornecem recursos avançados para processamento em um ambiente de nuvem. No entanto, nem todos os consumidores estão cientes da recolha e exportação destes dados, e as preocupações com a privacidade podem exceder o desejo do cliente pelas funcionalidades fornecidas. É importante que os consumidores tenham o controlo dos dados que fornecem fora das suas próprias redes, através de uma divulgação clara e de um processo de adesão, em vez de um processo de adesão por defeito.

10 Use apenas algoritmos criptográficos e modos de operação padrão da indústria para qualquer protocolo de segurança (como verificação de autenticidade de firmware)

Os algoritmos criptográficos são complexos e hoje em dia é razoável dizer que não existe uma pessoa capaz de dizer que qualquer algoritmo específico é seguro. A única maneira de ter confiança em um algoritmo é submetê-lo ao estudo de uma série de especialistas durante muitos anos. Mesmo assim, novas pesquisas e descobertas podem surgir e revelar falhas anteriormente não encontradas.

Portanto, é altamente recomendável que apenas algoritmos criptográficos, comprimentos de chave e modos de operação verificados por pesquisa e geralmente aceitos como seguros sejam usados. Uma ótima referência para isso é o NIST SP 800 57, que basicamente define Triple DES, AES, RSA e Elliptic Curve Cryptography (ECC) como os únicos algoritmos a serem usados.

O modo de operação, ou a forma como a criptografia é realmente usada para criptografar os dados ou fornecer autenticação, também é importante. Pode ser facilmente esquecido que o uso de um modo simples de operação, como o Livro de Código Eletrônico (BCE), pode realmente expor padrões nos dados de texto simples e pode não alcançar a segurança pretendida ao implementar a criptografia.



11 Forneça aos usuários a capacidade de ativar recursos sob demanda que eles podem não querer ou usar apenas de forma intermitente

Todo software possui bugs, e muitos desses bugs expõem possíveis vulnerabilidades de segurança. Quanto mais software um produto tiver, mais bugs ele provavelmente terá. É claro que esses bugs podem ser minimizados por meio de testes, correções e outros métodos, mas muitos bugs permanecerão não encontrados e sem correção até que um exploit seja lançado. Em contraste com isto, muitos produtos dependem de um extenso conjunto de recursos para se diferenciarem num mercado comercialmente agressivo. Para equilibrar esses requisitos conflitantes, é ideal desabilitar alguns dos recursos mais avançados por padrão, para que o sistema possa permanecer seguro mesmo se uma exploração for encontrada.

Por exemplo, acesso remoto, emparelhamento sem fio e funções avançadas (e-mail, interfaces de impressora, etc.) podem ser fornecidos, mas desativados por padrão e fornecidos com um recurso de acesso cronometrado para que o cliente possa ativar o recurso apenas por um determinado período.

12 Implemente um autoteste de inicialização que valide as funções principais e a integridade do firmware antes da execução. Implemente uma cadeia criptográfica de confiança do hardware durante a inicialização sempre que possível

Idealmente, o firmware deve ser validado em cada inicialização para garantir que não tenha sido alterado desde a instalação. De qualquer forma, isso pode ser alcançado quando há uma assinatura em todo o firmware, o que pode ser o caso se o sistema executar uma função executiva simples, mas é significativamente mais difícil quando se trata de um sistema operacional (SO) complexo, como o Linux. Validar todos os diferentes arquivos, scripts, etc. necessários para garantir que um sistema operacional complexo seja executado corretamente é complicado.

As soluções potenciais incluem a validação do carregador de inicialização do dispositivo (a partir de uma raiz de confiança de hardware, que requer suporte no processador que está sendo usado) e, em seguida, usar esse carregador de inicialização para validar uma imagem do sistema operacional que é descompactada e instalada.

Claro, tudo isso leva tempo. Mas é útil evitar coisas como Ransomware, que pode tentar instalar software que torna um dispositivo inoperante até que o valor do resgate seja pago.

13 Certifique-se de que quaisquer padrões do sistema, como senhas, certificados ou chaves, sejam forçados a serem alterados antes da operação inicial

Os padrões do sistema devem ser evitados sempre que possível, mas tais padrões são frequentemente necessários. Por exemplo, um padrão pode ser necessário para permitir a inicialização do sistema pela primeira vez. Isto pode ser aceitável, mas este valor padrão deve ser forçado a mudar como parte da configuração geral.

Em última análise, os padrões devem ser considerados apenas para certificados e outros itens que possam ser necessários para a operação normal, mas que devem ser alterados pelo usuário antes da instalação e operação. Isto também abrange quaisquer valores de teste que possam estar no firmware e nunca devem ser deixados em um sistema de produção.

14 Garanta que mensagens de erro ou respostas a mensagens inválidas não exponham dados confidenciais

Quando um sistema é acessado incorretamente, é comum retornar algum tipo de mensagem de erro para indicar qual erro ocorreu. Essas mensagens podem facilmente revelar informações confidenciais sobre um sistema e devem ser implementadas com muito cuidado. Considere retornar apenas indicações boas/ruins dos sistemas de produção, pelo menos até que alguma forma de estado de depuração seja ativada (através de um acesso autenticado, é claro). Nunca retorne detalhes de quaisquer dados descriptografados se houver algum tipo de falha, mesmo para observar qual erro ocorreu na descriptografia e validação. Simplesmente rejeite a conexão nesses casos.

15 Certifique-se de que as chaves criptográficas sejam usadas apenas para uma única finalidade

O gerenciamento de chaves (a forma como as chaves criptográficas são usadas) é extremamente importante. O algoritmo criptográfico é apenas uma parte da segurança. A forma como o algoritmo é usado e a forma como as chaves criptográficas necessárias são usadas também são vitais.

Para evitar comprometimento, é uma boa prática usar chaves criptográficas para apenas uma finalidade. As chaves de criptografia de dados devem ser usadas apenas para dados criptografados. As chaves usadas para proteger senhas, por exemplo, devem ser diferentes. Não misture chaves (ou pares de chaves) entre usos para criptografia e autenticação. Cada chave deve ter um uso exclusivo.

16 Implemente o menor privilégio em todos os sistemas

Os processadores modernos geralmente oferecem diferentes níveis de privilégio, que incluem acesso potencial à memória e outros recursos. Esses recursos podem ser usados pelo software para ajudar a proteger os ativos no dispositivo, garantindo que apenas o software com o privilégio correto possa acessá-los. A interface para os controles de privilégios de nível de hardware de um processador geralmente é gerenciada pelo sistema operacional, como o Linux, mas também pode ser controlada mesmo se um dispositivo não usar um sistema operacional complexo (por exemplo, se usar uma função executiva simples). , ou RToS reduzido).

Sempre que possível, reduza ao mínimo o uso de privilégios de root ou de nível de supervisor em sistemas embarcados e mantenha os dados secretos, como chaves criptográficas, no mais alto nível de privilégio (mais difícil de acessar). Regras semelhantes são válidas para aplicativos e sistemas baseados em nuvem: mantenha o uso de privilégios elevados ao mínimo e tente isolar funções como seu próprio usuário ou conjunto de privilégios.

17 Implementar proteções para impedir a execução de memória de dados

Muitos ataques remotos visam obter a execução de código fornecido pelo malfeitor, que é fornecido por meio de uma vulnerabilidade específica. É quase impossível evitar todas as vulnerabilidades no código, mas é possível mitigar o potencial de execução remota de código por meio de vulnerabilidades de codificação por meio de vários métodos diferentes.

Muitos processadores fornecem funções “no-execute”, que podem marcar áreas específicas da memória que não podem ser usadas para fazer referência a código executável diretamente. Alternativamente, alguns processadores podem até fornecer segmentos de código e memória de dados totalmente diferentes, o que torna impossíveis ataques de injeção direta de código (embora outros tipos de ataques ainda possam ser executados).

Além das proteções diretas de hardware, também existem outras proteções que podem ser aplicadas no nível de software. Alguns deles podem ser fornecidos pelo próprio sistema operacional e outros podem ser aplicados por meio das configurações do compilador ao criar o código-objeto para carregar no dispositivo.

Determine quais proteções são possíveis nos vários componentes de qualquer sistema e implemente tantas proteções quanto for técnica e operacionalmente viável.

18 Não permitir a execução direta de comandos, scripts ou outros parâmetros fornecidos externamente que não estejam dentro das funções definidas dos dispositivos

Além da execução direta de código, existem outras maneiras pelas quais um malfeitor pode obter acesso a um sistema se houver outros intérpretes ou ambientes de execução disponíveis. Por exemplo, um sistema pode permitir a execução de código não nativo, como JavaScript, o que pode levar a possíveis vulnerabilidades. Embora isso não signifique que seja sempre necessário desabilitar completamente coisas como JavaScript, vale a pena entender a necessidade do sistema por esse tipo de funcionalidade. Se for incluído, muitas vezes serão necessárias soluções de segurança mais complexas para manter a postura geral de segurança do sistema.

19 Crie e compile firmware para dispositivos para que contenha apenas código e sistemas necessários para as funções definidas. Sempre remova/desative recursos de depuração e desenvolvimento em dispositivos ao criar código de produção

Quanto maior o corpo do código, maior a chance de falhas de segurança não descobertas. Portanto, é prudente remover o máximo de código possível para limitar a chance de que uma falha descoberta após o envio do produto exija correção ou outras mitigações. Isso inclui código que pode não ser executado normalmente; mesmo que o código não seja usado, tê-lo no dispositivo pode levar a problemas de segurança no futuro.

Por motivos semelhantes, é essencial remover o código de depuração e desenvolvimento do dispositivo antes do envio. Isso geralmente é feito isolando instruções “ifdef”, que podem remover automaticamente esses recursos durante o tempo de compilação. Embora seja compreensível querer recursos no código caso haja problemas na produção, muitas vezes é mais provável que tais recursos se tornem uma fonte de vulnerabilidade, pois fornecem acesso e informações que normalmente não seriam fornecidas no ambiente do usuário final. .

20 Implemente um programa de gerenciamento de vulnerabilidades para monitorar e resolver regularmente falhas de segurança no produto antes do lançamento e até o fim da vida útil. Incluir um processo para distribuir patches aos clientes e mantê-los informados

A segurança é um alvo móvel e nenhum sistema será 100% seguro. À medida que novas vulnerabilidades e métodos de ataque são lançados, é importante ter um programa para garantir que os sistemas – novos e existentes – não sejam vulneráveis a estas falhas de segurança. Isto só é possível quando existe um programa aprovado e aplicado pela gestão para garantir a monitorização contínua e as atualizações da segurança do sistema.

A capacidade de instalar patches em sistemas não terá valor se os patches não forem criados e disponibilizados para instalação nos sistemas do cliente.

Para saber mais sobre os serviços de segurança cibernética da UL, visite UL.com/cybersecurity ou envie um e-mail para: ULCyber@ul.com

UL.com/cybersecurity